# UNIVERSIDAD CARLOS III DE MADRID

TRABAJO FIN DE GRADO


# SLAM 3D con datos RGB-D

*GRADO EN INGENIERÍA EN*

*TECNOLOGÍAS INDUSTRIALES*

Autor: Pedro Javier Guevara Fernández

Tutor: Luis Enrique Moreno Lorente

Leganés, 24 de Septiembre de 2014

# Resumen

La búsqueda de un robot completamente autónomo se ha considerado como uno de los temas con mayor potencial durante estos últimos años. El concepto de autonomía en términos de navegación robótica se traduce en ser capaz de obtener el recorrido óptimo para completar la tarea para la que se le ha programado, todo esto sin la disposición de información previa del entorno en el que debe trabajar, y por lo tanto sujeto a elevada incertidumbre.

El problema SLAM (Simultaneous Localization and Mapping) se aplica para resolver este tipo de situaciones. Mediante la combinación de técnicas de visión y algoritmos matemáticos, el robot puede trabajar en tiempo real sobre el terreno, actualizando el mapa disponible y adecuando la trayectoria según su posición en el mismo. Esta capacidad dota a este tipo de robots de la versatilidad requerida en diversas aplicaciones robóticas como misiones de exploración, rescate y mapeado de entornos, siendo este último el tema principal sobre el que se desarrolla este proyecto.

Los mapas tridimensionales proporcionan al robot completa información sobre el terreno en el que se mueve. Por otra parte, la gestión de mapas 3D implica una carga computacional que limita su utilización en ciertas situaciones. Por lo tanto, en este proyecto se propone como solución un consenso entre la funcionalidad de un mapa topológico en dos dimensiones y la utilidad de la información proporcionada por el mapa 3D.

Mediante el procesamiento individual de los modelos 3D de varias habitaciones de una vivienda y su posterior integración en un mapa topo-geométrico global de la misma, esta propuesta de mapeado combina una gestión constante del mapa global 2D con la capacidad de acceder a la representación tridimensional de la estancia que el robot ocupa en ese momento.

# Abstract

The search for completely autonomous mobile robots has been considered one of the main robotic topics over the last years. Autonomy in terms of robot navigability is translated into being able to obtain the most suitable paths in order to complete the assigned tasks with the absence of an initial input data, and so being subject to a great amount of uncertainties.

The Simultaneous Localization and Mapping (SLAM) problem is applied to solve this issue. Through a combination of vision techniques and mathematical algorithms, the robot can update any environment map and process the most suitable path considering its own position within it. The online operation of this method grants the versatility required for many different robotic applications such as exploration, rescue missions and map building techniques, being this last one covered in the scope of this project.

Tri-dimensional maps give all the possible information required for a robot to perform any given task. On the other hand, the computational consumption represents the highest disadvantage of managing this type of maps. Therefore, a consensus between the functionality of topological 2D maps and the utility of 3D models is proposed as a solution in this project. By processing individual 3D models associated to every room in a real house and through the integration of these models in a global topo-geometric map, this mapping proposal combines the constant management of the global map with a detailed approach on the tri-dimensional representation of the room it´s currently occupying.

# Índice de contenido

## Contenido

# Índice de ilustraciones

# Índice de tablas

# Listado de Acrónimos

| | | |
|---|---|---|
| **SLAM** | : | Simultaneous Localization and Mapping |
| **RGB-D** | : | Red/Green/Blue-Depth |
| **RANSAC** | : | Random Sample Consensus |
| **ICP** | : | Iterative Closest Point |
| **g2o** | : | General Graph Optimization |
| **PCL** | : | Point Cloud Library |
| **ROS** | : | Robot Operative System |
| **UAV** | : | Vehículo aéreo no tripulado |

# 1. Introducción

El empleo de robots móviles para realizar tareas imposibles o no deseadas por el ser humano es un tema que ocupa actualmente uno de los puestos más destacados en el desarrollo tecnológico a nivel global. Si bien existe una moda alrededor de los robots antropomórficos, la realidad es que la búsqueda de total autonomía se puede aplicar a toda la variedad de robots con capacidad de movimiento.

Una de las características más importantes con las que un organismo (artificial o no) debe contar para poder considerarse completamente autónomo es la capacidad de poder ubicarse, con la consiguiente posibilidad de poder tomar decisiones basadas en la posición ocupada dentro de un entorno más o menos extenso.

Hay ocasiones en las que es posible proporcionar al robot una representación más o menos exhaustiva del entorno en el que se encuentra, y el usuario tiene el control suficiente como para colocar a este robot en una posición inicial. En este caso, con conocer el movimiento propio, un robot sería capaz de conocer su posición en cualquier momento.

Sin embargo, la disposición de un mapa documentado es una opción que para las aplicaciones más prometedoras y sofisticadas orientadas a robots móviles no va a estar disponible. La exploración espacial, las operaciones de rescate en entornos peligrosos, etc. son ejemplos de actividades para las que los robots móviles resultarían increíblemente útiles y para las cuales, y por razones obvias, éstos no van a contar con un mapa representativo de la zona en la que van a operar.

Con un énfasis especial en la exploración espacial (con la relativamente reciente puesta en marcha del proyecto Curiosity en la superficie de Marte), el marco socio-económico en el que surge el problema anteriormente planteado nos permite a su vez obtener una solución: el SLAM, que si bien para aplicaciones muy sofisticadas requiere el empleo de equipos y técnicas más potentes, se puede aplicar a un nivel más local de forma relativamente sencilla.

Con el desarrollo de software libre, la continua actualización y optimización de éste y la colaboración de la comunidad alrededor de los incontables proyectos existentes es posible encontrar una configuración de software adecuada tanto para los experimentos más exhaustivos como para un estudio más sencillo del método SLAM como seria el constituido por este proyecto.

## 1.1 Objetivo del proyecto

El objetivo de este proyecto es la utilización del concepto SLAM para generar un mapa tridimensional de un entorno controlado, de manera que se pueda utilizar posteriormente para usos tales como navegación de robot y similares.

Inicialmente se estableció como objetivo del proyecto conseguir una compatibilidad entre cámaras comerciales y MatLab para realizar el proceso de SLAM de diversos entornos. Hay numerosas aplicaciones y código generado. Se realizó un estudio de la compatibilidad de las cámaras disponibles con diversos paquetes asociados a MatLab, pero a la hora de identificar una línea de trabajo para aplicar SLAM a los datos recogidos por las cámaras mencionadas, se vio cierta limitación a la hora de emplear esta plataforma.

Durante este paso, se vio que era más realista y práctico emplear un software libre diseñado específicamente para realizar SLAM, habiendo multitud de opciones en la web y con extensa documentación en cada caso, dejando abiertas mayores posibilidades para encontrar aplicaciones al proyecto.

Como resultado de la colaboración de la comunidad previamente mencionada fue relativamente sencillo dar con RGBD-SLAM y ponerse en contacto con Felix Endress, de la Universidad de Freibourg, uno de los principales responsables del desarrollo de este software en particular, con el que se discutió la posibilidad de utilizar una versión todavía en fase de pruebas diseñada para la última actualización de Ubuntu y ROS con el objetivo de proporcionar “feedback” de gran utilidad para la optimización del proyecto RBGD-SLAM. Finalmente se optó por una versión más antigua pero estable, diseñada para ROS-fuerte (una versión menos reciente y por lo tanto más consolidada del sistema operativo ROS, que se detallará más adelante), seleccionada por la seguridad que ofrecía con relación a la obtención de resultados.

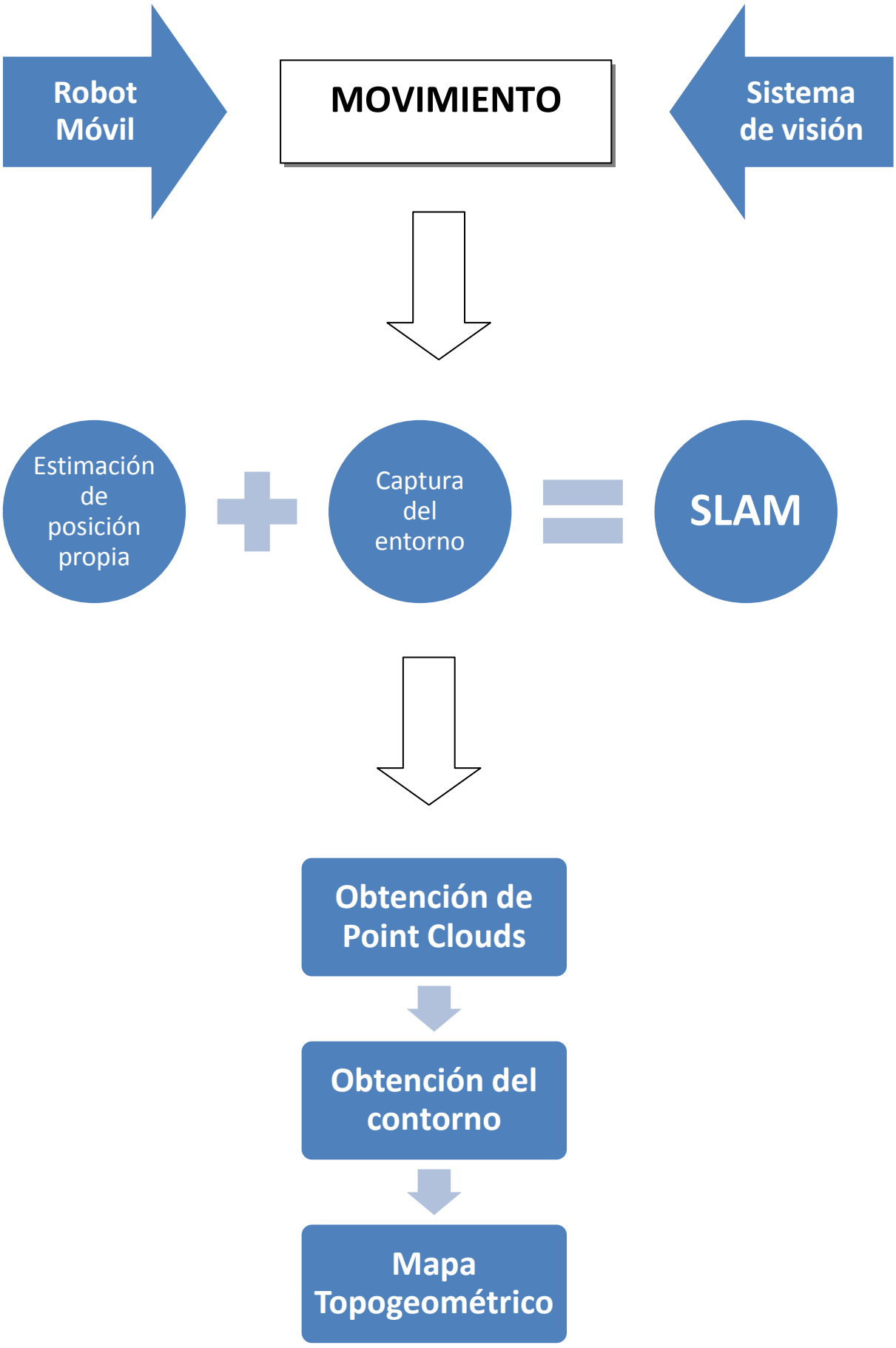
Una vez establecida la forma de obtener los datos, el objetivo del proyecto ha sido la generación y manipulación de un mapa de cara a la utilización de éste por parte de un robot móvil, buscando siempre un equilibrio entre la exactitud y precisión y la carga de cálculo y practicidad del método empleado.

Para la consecución del objetivo final, se van a seguir una serie de pasos o etapas que se detallarán a continuación:

1. Selección del entorno a mapear.
2. Captura y obtención de las nubes de puntos representativas de todas las estancias o elementos necesarios del entorno elegido.
3. Procesamiento de las nubes de puntos obtenidas con el fin de permitir una manipulación posterior. Dicho procesamiento estará constituido por operaciones de :
    a. Filtrado
    b. Suavizado de superficies (opcional)
    c. Extracción de planos representativos
    d. Extracción de los contornos de los planos representativos
4. Como información adicional, se obtendrán los datos asociados al movimiento de la cámara durante la captura.
5. Construcción de un mapa topogeométrico con los contornos extraídos.

La elaboración de la memoria a su vez va a seguir una estructura que tiene el fin de facilitar el entendimiento de los procedimientos aplicados:

- Objetivo del proyecto
- Concepto de SLAM
    - Estado del arte
    - Principios del problema
    - Funcionalidad
- Hardware utilizado
- Software utilizado
    - RGBDSlam
        - Funcionamiento del software
        - Algoritmos empleados
        - Parámetros obtenidos (point clouds, estimaciones de trayectorias,etc)
    - Point Cloud Library
        - Explicación del funcionamiento básico del programa
    - Cloud Compare
        - Funcionalidad utilizada
- Planteamiento del problema y variantes
- Resultados obtenidos
- Conclusiones y análisis de resultados
- Valoración del proyecto
- Trabajo futuro


Robot Móvil
MOVIMIENTO
Sistema de visión
Estimación de posición propia
Captura del entorno
SLAM
Obtención de Point Clouds
Obtención del contorno
Mapa Topogeométrico

## 1.2 Plano de la vivienda

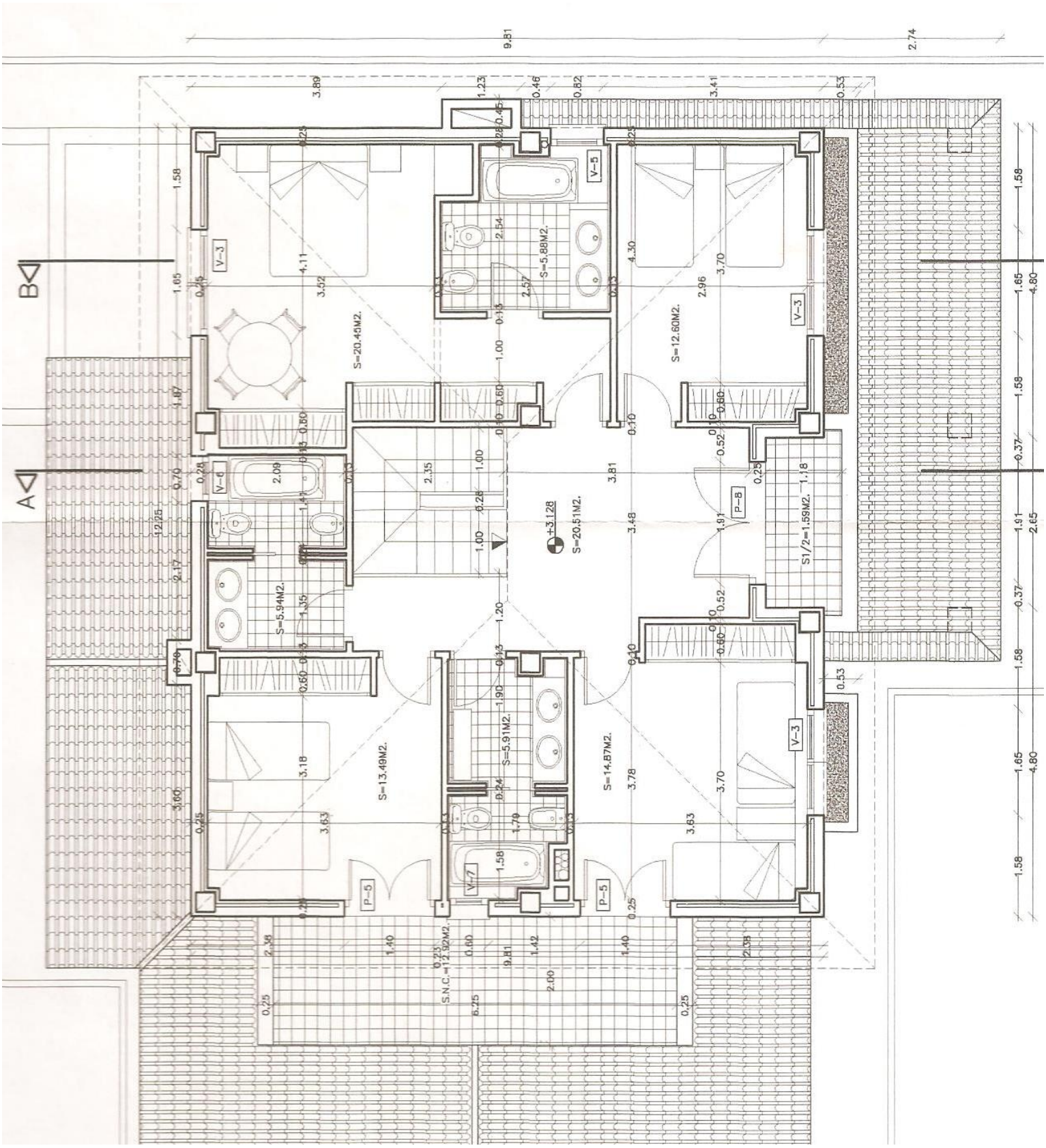
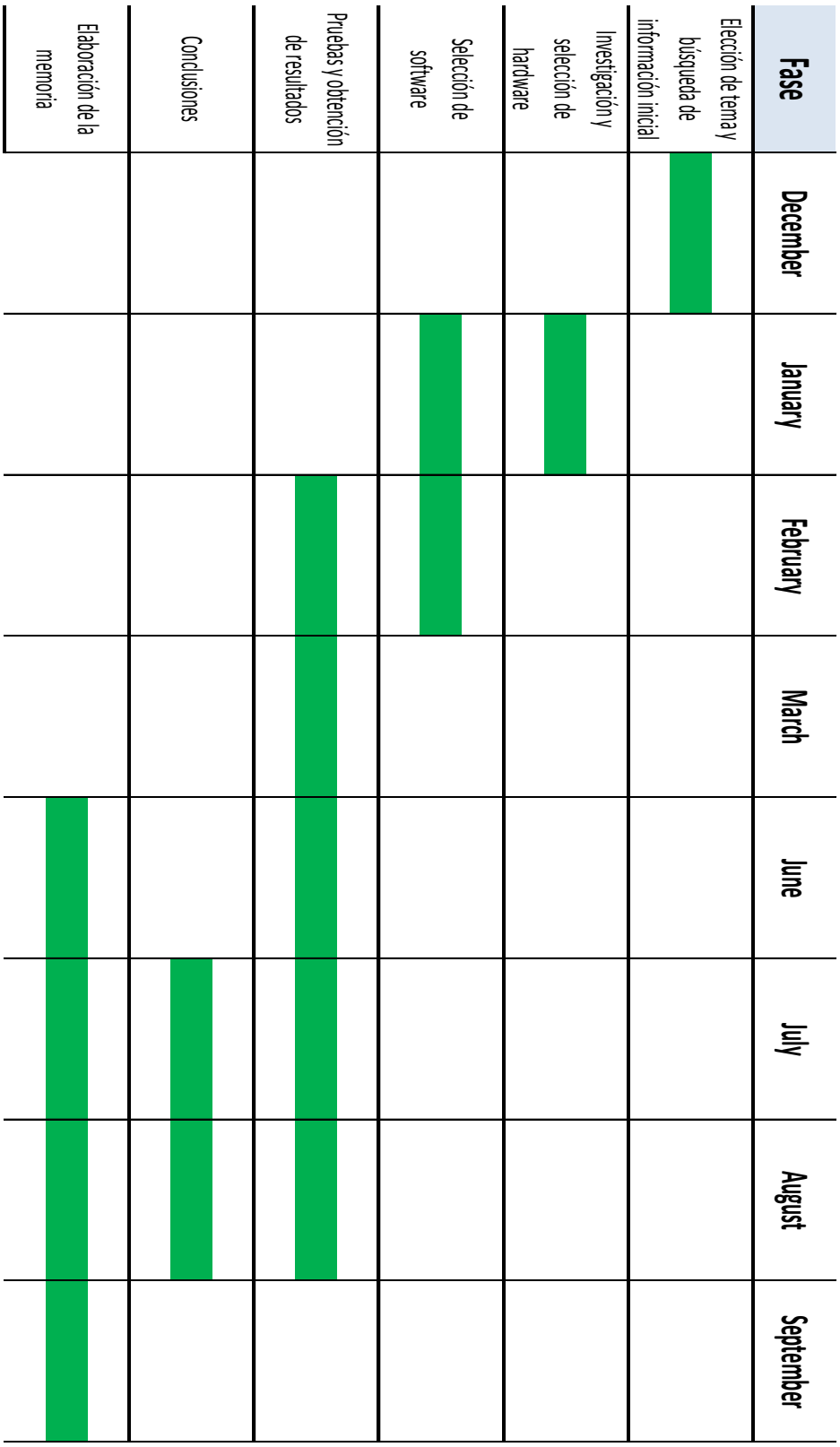



**Ilustración 1 – Plano de la vivienda utilizada n el proyecto**

## 1.3 Planificación del proyecto



# 2. SLAM

El concepto Simultaneous Localization And Mapping surge de la necesidad de resolver el problema del "huevo y la gallina" que nos encontramos al intentar dotar de autonomía a un robot móvil en un entorno desconocido.

Para poder generar un mapa debemos saber donde nos encontramos, y para saber dónde estamos necesitamos basarnos en un mapa. El método SLAM realiza ambas operaciones de forma simultánea mediante procesos iterativos, generando un mapa del entorno y cotejando la posición propia con dicho entorno.

Si bien el paso previo al proceso SLAM pasaba por la captura de datos mediante dispositivos láser, convirtiéndolo en una tarea al alcance de personal con el material especializado adecuado, la aparición de cámaras comerciales que cumplen dicha funcionalidad ha permitido realizar grandes avances en este campo, apareciendo en estos últimos años numerosos ejemplos de aplicaciones y estudios sobre SLAM

El proceso SLAM está constituido por una serie de elementos o pasos que en conjunto permiten proporcionar al robot los datos requeridos:

- Captura de datos por medio de un sensor de profundidad
- Captura de datos del movimiento propio
- Nodos de referencia
    - De la posición propia del robot
    - Elementos representativos del entorno
- Algoritmos de asociación de datos

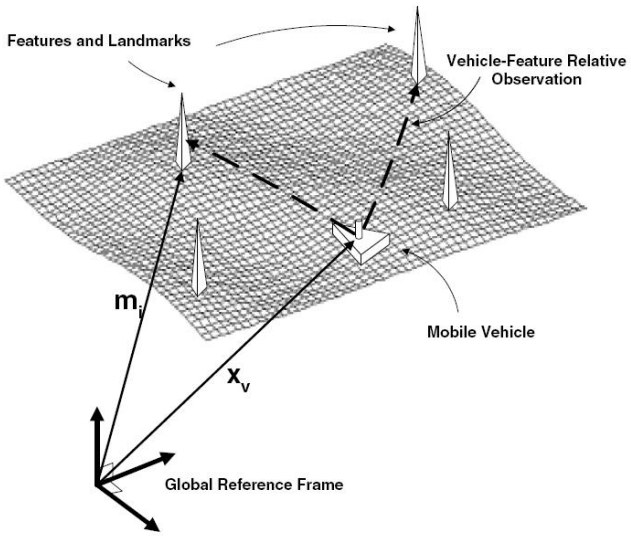



Ilustración 2 – Presentación gráfica general del problema SLAM

## 2.1 Estado del arte

En la actualidad, hay mucho interés en el desarrollo y mejora del SLAM, que aun considerado un problema resuelto a nivel teórico y conceptual, se encuentra en un proceso de búsqueda de optimización constante.

Hay un rango de aplicaciones consideradas para el SLAM y orientada a diversas tareas programadas en robots móviles:

- Operaciones en interior
- Operaciones en exterior
- Submarinas
- Subterráneas

¿Por qué es tan complicado aplicar SLAM?

El problema de las incertidumbres y errores es una fuente de preocupación en cualquier aplicación de robots móviles. Sin embargo, es especialmente grave en el caso del SLAM.

La asociación constante que debe producirse entre la información sobre el movimiento relativo del robot respecto a un punto de referencia propio y el movimiento absoluto con respecto al mapa global con el que trabaja el método SLAM hacen que los algoritmos utilizados sean lo suficientemente robustos como para minimizar el error cometido.

Como se ha mencionado previamente, la aparición de cámaras disponibles ha convertido el SLAM en un problema que puede ser estudiado por un número cada vez más grande de personas y con nivel de profundidad ajustado según la aplicación. Este incremento en las líneas de investigación ha generado un crecimiento en la cantidad de experimentos y estudios publicados durante los últimos años.

Hay que entender que el concepto de SLAM como tal comprende una definición básica de funcionalidad y resolución del problema de generación de mapas tridimensionales y localización del robot dentro del mismo. Por otra parte, el conjunto de algoritmos, operaciones y pasos aplicados para conseguir dicho objetivo es variable, y hay numerosas alternativas dependiendo de la aplicación y las condiciones de trabajo del robot.

## 2.2 SLAM probabilístico

El concepto de SLAM probabilístico surge de la formulación del problema previamente mencionado en término de las siguientes variables:

$X_t$ : Posición real del robot en un tiempo t, referido a un sistema de coordenadas total

$U_t$ : Desplazamiento del robot entre la posición Xt-1 y Xt, referido a un sistema de coordenadas global

$m_i$ : Posición del punto de referencia i-ésimo, referido a un sistema de coordenadas global

$Z_{i,t}$ : Posición del punto de referencia i-ésimo en el instante t, referido al sistema de referencia propio del robot.

La teoría del SLAM probabilístico podría resumirse con:

"La probabilidad de determinar la posición real del robot en el instante t, así como la de un punto de referencia asociado siendo conocidos la posición inicial del robot en el instante t=0, el desplazamiento total desde el origen en el instante t y la estimación del elemento de referencia con respecto al propio robot en el instante t"

$$P\ (X_t, m \mid Z_{0;t}, U_{0;t}, X_0)$$

Es aquí donde entran en juego las dos estimaciones que el robot debe hacer a la hora de poder resolver el problema SLAM

- Probabilidad de realizar una observación correcta de un elemento de referencia dada la posición del robot y la posición del elemento de referencia.

$$P\ (Z_t \mid X_t, m)$$

- Probabilidad de estimar la posición del robot dada la posición anterior de este y la variación de desplazamiento.

$$P\ (X_t \mid X_{t-1}, U_t)$$

De estas ecuaciones se obtiene tanto una presentación de la solución al problema como la evidente fuente de errores derivados de la incertidumbre del cálculo de la posición propia del robot, que se traduce en una incertidumbre en el parámetro Xt de la estimación del elemento de referencia.

### 2.2.1 Localización

Esta primera parte del problema surge de la diferencia entre la señal del control enviada al robot (los actuadores del robot) y el verdadero desplazamiento que este realiza.

- El robot parte de una posición $X_o$, con una serie de puntos de referencia localizados desde esa posición
- Posteriormente, realiza el desplazamiento programado y en vez de acabar en una posición $X_1$, su recorrido finaliza en una posición $X_1'$.
- El robot, asumiendo encontrarse en la posición $X_1$, ha previsto encontrar los puntos de referencia iniciales (denominados $m_1$ y $m_2$) en unas posiciones $Z_1$ y $Z_2$.
- Sin embargo, debido al error de desplazamiento, encuentra que dichos puntos de referencia están situados en unas posiciones $Z_1'$ y $Z_2'$.
- El problema de localización se resuelve utilizando estos elementos de referencia para estimar la verdadera posición actual del robot $X_1'$.

Al prolongar el recorrido y recibir subsecuentes señales de control, la estimación de la posición propia va disminuyendo su incertidumbre, ya que se van cotejando los elementos de referencia desde posiciones distintas, y estos permanecen inamovibles en el tiempo.

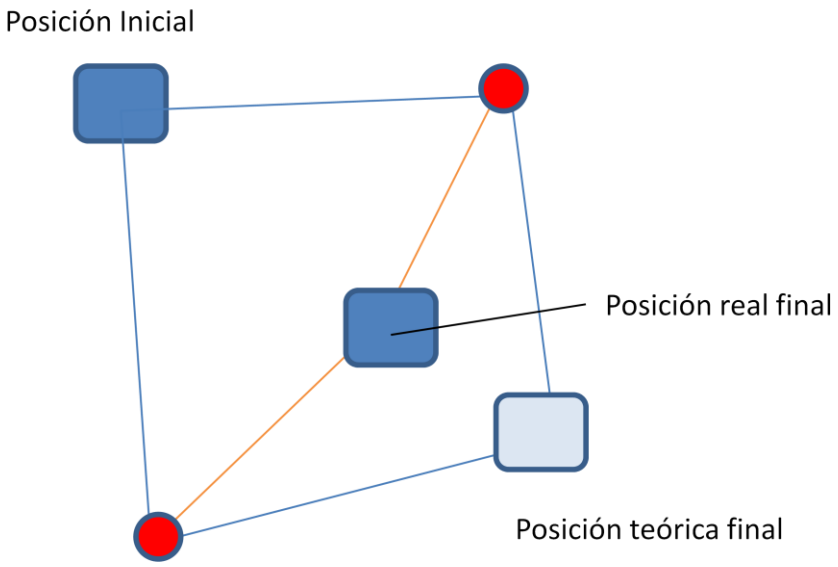



Ilustración 3 - Desplazamiento real vs teórico

### 2.2.2 Mapeado

En esta sección del problema SLAM, la diferencia radica en que se asume la total precisión en la posición propia y se presupone una incertidumbre en los puntos de referencia o puntos de interés de los elementos observables.

- El robot parte de una posición Xo, con una serie de puntos de referencia localizados desde esa posición.
- Posteriormente, realiza el desplazamiento programado hasta una posición X1 y realizar una captura visual del entorno.
- Debido a imprecisiones en la medida, los elementos de referencia son registrados en una posición que no es la real con respecto al robot.
- El robot realiza un movimiento posterior, situándose en una posición X2 y registrando otra vez a su alrededor en búsqueda de elementos de referencia.
- Desde esta nueva perspectiva y con el fin de continuar el proceso de mapeo, el robot cotejará los datos obtenidos previamente para un mismo punto de referencia y considerará la variación de ambos puntos, producto de la incertidumbre en cada una de las medidas tomadas en X0 y X1.
- Aplicando un proceso de minimización de los errores (el algoritmo puede variar) se actualiza el mapa desde la posición X2 teniendo en cuenta todas las perspectivas utilizadas previamente.

Para realizar estos cálculos, debe asumirse un sensor calibrado, por lo que la naturaleza de los errores cometidos en las medidas es hasta cierto punto constante.

### 2.2.3 Simultaneidad

Una vez introducidos los problemas por separado, se observa la imposibilidad de unificar ambas soluciones sin eliminar los parámetros de entrada que se suponen conocidos y exactos en cada caso.

El algoritmo de localización necesita una percepción de elementos de referencia exacta (o lo que es lo mismo, un mapeado exacto), mientras que el algoritmo de mapeado precisa de un seguimiento de la localización del robot exacta también.

Un aspecto que surge a la hora de combinar ambos procesos es la necesidad de priorizar uno de los dos algoritmos sobre el otro. De manera que si se realiza el mapeo previo a la estimación de la posición propia, este último resultado se vería más favorecido en términos de exactitud que si hiciésemos la operación en orden inverso, donde sería la tarea de mapeado la que tomaría mayor importancia.

### 2.2.4 SLAM front-end

Es la parte encargada de procesar todos los datos provenientes de los distintos sensores (tanto relacionados con la odometría del problema como con los relacionados con la visión y el escaneado del entorno). En este proceso se tienen en cuenta:

1. Localización y posicionamiento del robot
2. Detección de bucles
3. Asociación de datos del escáner

### 2.2.5 SLAM back-end

Esta parte esté relacionada con la computación de los datos obtenidos en el SLAM front-end. Aquí entran en juego los algoritmos de optimización y computación, así como los tratamientos de error que hiciesen falta.

### 2.2.6 Concepto de cierre del bucle

El cierre del bucle en el concepto SLAM se produce al revisitar un área previamente mapeada.

La principal característica del cierre del bucle es el colapso de las incertidumbres en el mapeado, haya preciso este cierre o no, tanto en los puntos de referencia como en la posición relativa estimada del robot.

### 2.2.7 Nodos

Se ha nombrado el concepto de nodo varia veces durante la explicación del problema SLAM. Sin embargo, es mucho más sencillo relacionar el concepto de nodo en términos de la localización y el mapeado con una serie de resultados experimentales que lo soporten.

Hemos definido un nodo como cada uno de los instantes t en el grafo asociados a un registro de actualización de las variables internas del problema:

- Localización propia estimada
- Mapa estimado asociado a esa localización

De forma práctica, podemos ver la correspondencia entre estas dos variables utilizando el ejemplo previo correspondiente a un ejemplo de aplicación de la solución global al mapeado del entorno.

Si cogemos un nodo aleatorio (en este caso vamos a intentar seleccionar un nodo asociado a una de las regiones correspondientes a uno de los dormitorios), tendremos asociado un punto del grafo de trayectoria.

Del resultado de mover el sensor por una región controlada de la vivienda, se ha obtenido la siguiente distribución de nodos en el proceso SLAM:

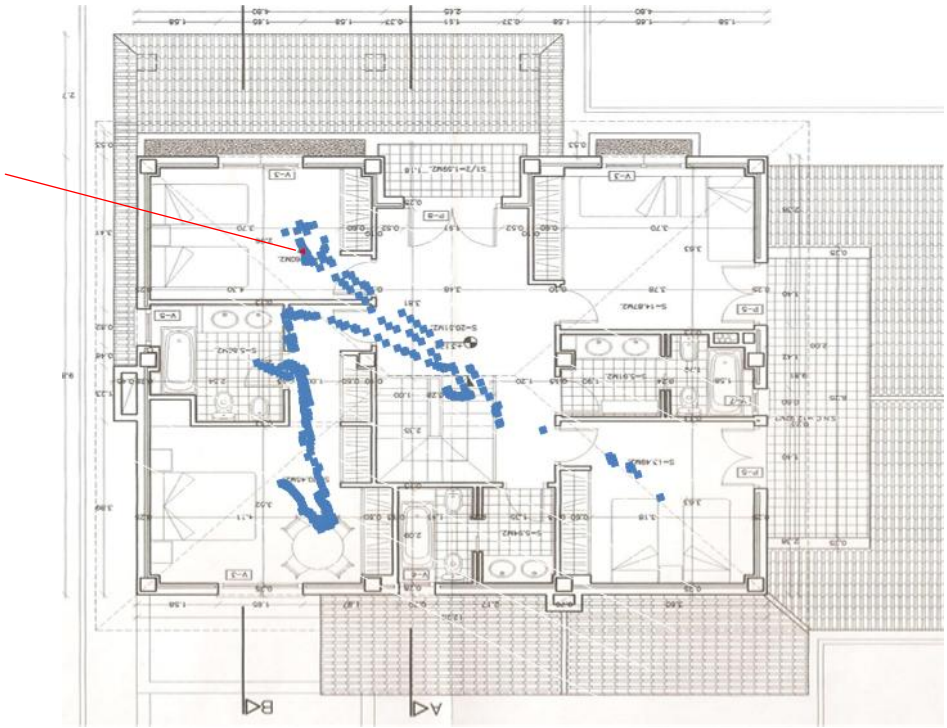


Ilustración 4 - Nodos asociados a una trayectoria

Asociado al nodo número 87 (marcado en rojo), vemos que este tiene como variables:

| Referencia Temporal | Coordenada X | Coordenada Y | Coordenada Z | q1 | q2 | q3 | q4 |
|---|---|---|---|---|---|---|---|
| 1410885415,59751 | -1,215962 | 4,596848 | 0,38638 | 0,016317 | -0,203359 | 0,401701 | 0,892757 |

Tabla 1 - Coordenadas XYZ y cuaternios asociados a un nodo

Observamos que el nodo cuenta con toda la información concerniente a la posición y a la orientación del dispositivo en ese instante, parámetros indispensables como ya se ha indicado en la introducción del problema SLAM a la hora de estimar la posición propia.

Esta representación nos proporciona información interesante también de cara a la interpretación de los algoritmos de detección de características/puntos de interés utilizados por el software, ya que vemos de forma visual los elementos que el robot ha considerado como representativos de la escena a la hora de registrar las referencias visuales.

Se adjunta el ejemplo de fotograma asociado al nodo 87.

Ilustración 5 - Fotograma asociado al nodo 87

## 2.3 Aplicaciones SLAM

El rango de aplicaciones de SLAM en el mundo real es muy extenso.

Actualmente se conducen numerosas líneas de investigación para obtener los sistemas SLAM más apropiado para cada uso.

Uno de los ejemplos más importante de los avances realizados en la materia se ve representado por el DARPA Robotics Challenge, que en cada edición propone una aplicación asociada a la autonomía de los robots tales como:

- Recorrido en tiempos establecidos
- Navegación por entornos urbanos
- Robots orientados a tareas de mantenimiento de emergencia

Otras aplicaciones estudiadas consisten en la búsqueda de un mapeado fiable de entornos hostiles para el ser humano. Se realizan avances con la finalidad de obtener representaciones de cuevas y minas inaccesibles.

Los ambientes submarinos también constituyen un entorno de estudio apropiado para la aplicación del problema SLAM.

Además, se está buscando aplicar el concepto SLAM para operar robots aéreos (UAVs), destinados a distintas misiones de búsqueda y rescate. El proyecto ICARUS (Integrated Components for Assisted Rescue and Unmanned Search operations) constituiría un perfecto ejemplo de estas iniciativas.


**Ilustración 6 - De izq. a decha;: 1) Coche autónomo,Stanford 2)Robot Guía de Museo, actualiza automáticamente el mapa del recinto, Toyota. 3)"Omnirob" basado en KUKA y que opera con la sola ayuda de sus propios sensores**

### 2.3.1 Tipos de mapas generados en SLAM

- Mapa de rejilla

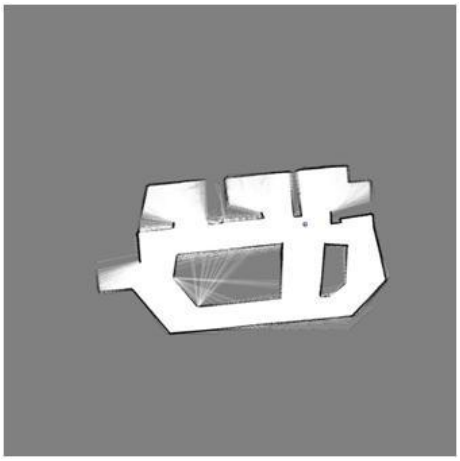


Ilustración 7 - Mapa de rejilla de la planta de un edificio

- Mapas basados en elementos de referencia


Ilustración 8 – Colocación de elementos de referencia en pista de tenis

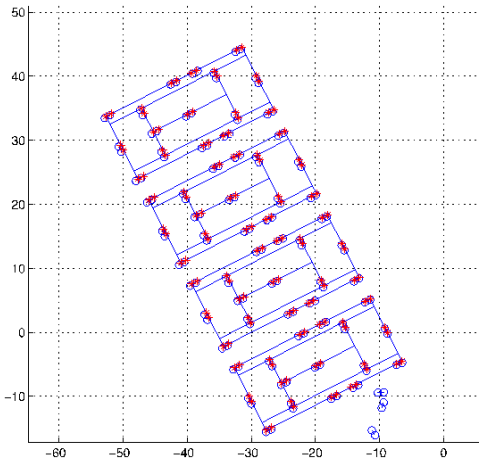



Ilustración 9 – Generación de mapa (Landmarks resaltados en rojo)

## 2.4 Mapa topogeométrico

Una vez obtenido el modelo 3D del SLAM, se ha establecido el objetivo de utilizar las representaciones locales de cada estancia para generar un mapa topogeométrico de la planta.

Para realizar esto, primero deberemos agrupar todas las nubes procesadas y combinarlas de manera que guarden relación con la distribución de las habitaciones en el modelo real de la vivienda.

Una vez distribuidas, a cada estancia se le va a asignar un nodo de referencia de la estancia con respecto al entorno global. Así mismo, el procedimiento va a consistir en la generación de una serie de vectores de distancia entre nodos, así como de un método de correspondencia entre los modelos locales, indicando al robot de manera matemática la disponibilidad de una trayectoria directa entre la estancia actual y la estancia a la que desea desplazarse.

### 2.4.1 Matriz topogeómetrica

Vamos a designar como matriz topogeométrica a aquella matriz que proporciona la información necesaria para comprobar la viabilidad de una trayectoria directa entre los distintos entornos locales (habitaciones o estancias) disponibles.

Una primera aproximación consistiría en asociar a cada estancia una columna y una fila de dicha matriz, de manera que se generaría una matriz cuadrada del tipo:

| | Modelo 1 | Modelo N |
|---|---|---|
| Modelo 1 | $\cdots$ | |
| | $\vdots$ | $\ddots$ $\vdots$ |
| Modelo N | $\cdots$ | |

Al cruce entre cada $\text{Modelo}_i$ y $\text{Modelo}_j$ se le asignaría un 1 o un 0 dependiendo de si cuentan con una correspondencia directa entre ellos.

De esta forma, el robot es capaz de interpretar las distintas posibilidades de movimiento disponiendo de esta matriz.

Sin embargo, una vez propuesto este modelo para la correspondencia entre estancias del mapa topogeométrico, se podría establecer otro criterio, más complicado, que permita al robot conocer no solo los entornos locales inmediatamente accesibles y ocultando la información al respecto de las estancias no accesibles, sino que se podría asignar un número dependiendo del número de cambios de habitación que se deban realizar para la tarea de desplazarse desde el Nodo i al nodo j.

Este ejemplo se puede observar en la siguiente ilustración:

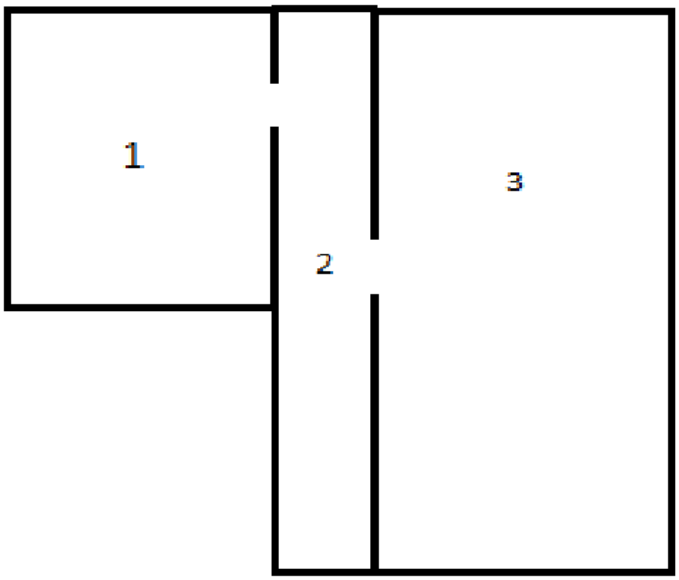



Ilustración 10 – Ejemplo de mapa topogeométrico con asignación de secciones

Si tuviésemos un robot situado en la habitación correspondiente al Nodo 1 del mapa topo geométrico, la matriz asociada a dicho nodo sería la siguiente

$$\begin{matrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{matrix}$$

Se asigna el número 0 a la correspondencia de una estancia consigo misma.

La matriz asociada a un nodo siempre va a ser:

- Simétrica

  Esto es debido al hecho de que la correspondencia entre los "saltos" a realizar entre 2 estancias es igual en ambos sentidos

- Diagonal igual a 0

  Considerando la distribución de las filas y columnas (Fila para el Nodo 3 = Columna del Nodo 3), esta propiedad se va a cumplir siempre.

- Cuadrada

Por otra parte, teniendo en cuenta la representación bidimensional del mapa topogeométrico, es posible asociar una serie de vectores de desplazamiento entre los diferentes nodos. Este vector puede estar definido para partir del centro de una estancia y realizar una conexión con el centro de otra cualquiera.

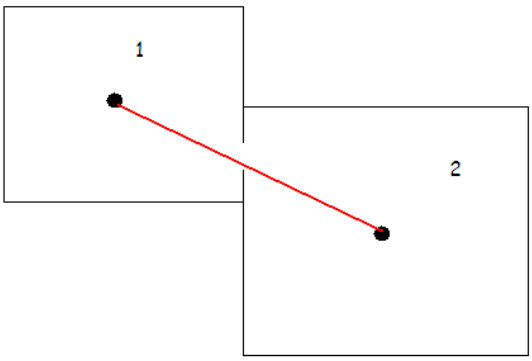



Ilustración 11 - Vector de unión entre nodos del mapa topográfico

Cada uno de estos puntos de referencia debe de tener asignada una posición con respecto al sistema de coordenadas global.

# 3. Recursos empleados

### 3.1.1 Sistema operativo

Se ha trabajado en Ubuntu 12.04 LTS

### 3.1.2 Entorno y lenguaje de programación

Para la parte de programación del proyecto se ha utilizado:

- C++ como lenguaje de programación
- Geany como entorno de programación
- CMake como herramienta de compilación y generación de ejecutables

## 3.2 Hardware

Microsoft Kinect


Ilustración 12 . Sensor Microsoft Kinect

Asus Xtion


Ilustración 13 – Sensor ASUS Xtion Pro Live

Estas cámaras vienen dotadas con un sensor de profundidad así como de un sensor RGB, que nos permite almacenar la información de las Nubes de Puntos con una componente de color, que si bien no es estrictamente utilizada en este proyecto en particular, permite utilizar este mismo material para diversas futuras aplicaciones.

### 3.2.1 Funcionamiento de la cámara

Tabla 2 - Especificaciones de la cámara ASUS Xtion Pro LIVE

| Especificación | Descripción |
|---|---|
| Power Consumption | below 2.5W |
| Distance of Use | Between 0.8m and 3.5m |
| Field of View | 58° H, 45° V, 70° D (Horizontal, Vertical, Diagonal) |
| Sensor | RGB& Depth& Microphone*2 |
| Depth Image Size | VGA (640x480) : 30 fps<br>QVGA (320x240): 60 fps |
| Resolution | SXGA (1280*1024) |
| Platform | Intel X86 & AMD |
| OS Support | Win 32/64 : XP , Vista, 7, 8<br>Linux Ubuntu 10.10: X86,32/64 bit<br>Android(by request) |
| Interface | USB2.0/ 3.0 |
| Software | software development kits(OpenNI SDK bundled) |
| Programming Language | C++/C# (Windows)<br>C++(Linux)<br>JAVA |
| Operation Environment | Indoor |
| Dimensions | 18 x 3.5 x 5 cm |

Se decidió por la utilización del modelo ASUS debido a que no requiere fuente de alimentación independiente de la conexión USB al ordenador. Esta característica le da unas ventajas en términos de portabilidad y autonomía que sobrepasan notablemente las de la Microsoft Kinect.

## 3.3 SOFTWARE

RGBDSLAM es un software diseñado por investigadores en la Universidad de Freibourg. Constituye un paquete o "package" (a los que haremos alusión secciones posteriores) integrado en ROS. Es por lo tanto imposible hacer una valoración de RGBDSLAM sin estudiar primero el funcionamiento y la organización de ROS, ya que en última instancia el software acaba sirviéndose de numerosas características y librerías incluidas en éste.

### 3.3.1 ROS

Ros (Robot Operative System) supone un paso adelante en el desarrollo de programas y aplicaciones en robótica. Desarrollar software orientado a robots es una tarea muy complicada para ser llevada a cabo por un solo individuo. ROS nace de la necesidad de la cooperación entre distintos grupos de expertos e investigadores con el fin de buscar las mejores soluciones para problemas actuales. Mediante un extenso sistema de herramientas, librerías y detallada documentación, ROS consigue coordinar el esfuerzo de diversos grupos de desarrollo que pueden adaptar soluciones de otros proyectos y mejorarlas o implementarlas en sus propios proyectos, sirviendo de base para que mediante el fruto de su trabajo el proceso se vuelva a repetir.

Este sistema operativo funciona por medio de "packages" (una muestra de la modularidad de este sistema, que puede adecuarse exactamente al conjunto de funcionalidades que el usuario busca) que se van añadiendo según la necesidad del usuario. RGBDSLAM constituiría uno de estos paquetes. Si bien la curva de aprendizaje de ROS para utilizarlo con todo su potencial es muy elevada, en este proyecto se proporciona una guía de instalación simplificada, para que un usuario no familiarizado con el sistema operativo pueda realizar los procesos necesarios para reproducir los resultados obtenidos en el trabajo.

ROS cuenta con una amplia variedad de librerías para utilizar, así como de todos los drivers y componentes necesarios para hacer que la toma de datos sea factible, y la compatibilidad con cualquier cámara utilizada sea total.

### 3.3.2 RGBDSLAM

#### *3.3.2.1 Frontend*

El procedimiento seguido por el Software se puede resumir en la siguiente serie de pasos:

Se obtienen imágenes de profundidad y color del sensor utilizado. De cada fotograma se extraen una serie de puntos de referencia o "feature descriptors", que constituyen los puntos más representativos del fotograma considerado.

Una vez aislados dichos puntos, se cotejan con los puntos más representativos de imágenes anteriores para analizar la transformación y la posible asociación del fotograma actual con los anteriores.

La solución más exacta sería la de realizar el proceso de comparación con todos los fotogramas previos. Sin embargo, RGBDSLAM se caracteriza por buscar una velocidad de procesamiento rápida, por lo que se seleccionan:

- Los 3 fotogramas previos al actual
- Un conjunto de 20 fotogramas aleatorios de los almacenados desde el inicio de la captura de puntos

Para realizar el estudio de la posible transformación entre fotogramas, se aplicar el algoritmo RANSAC. Como parámetro de entrada al algoritmo (cuyo funcionamiento se detalla más adelante) se seleccionan 3 parejas de puntos candidatos entre los 2 fotogramas que se encuentran en proceso de comparación en cada momento. Tomando como referencia estas 3 parejas, se aplica la transformación a todos los puntos definidos como "feature descriptors". Si la correspondencia entre todos los puntos más representativos es suficientemente buena, se acepta la asociación de ambos fotogramas y se procede a optimizar la transformación entre ambas.

Al aceptar que el fotograma actual corresponde a uno o más de los fotogramas predecesores, este se añade como un nodo más a computar por el SLAM "backend".

En caso de no encontrarse ningún predecesor, el método SLAM podría seguir 2 alternativas:

- Almacenar la posición actual como nodo y comenzar el mapeo de nuevo en otra región del espacio, esperando cerrar el bucle eventualmente
- Asumir que el nuevo nodo está asociado al nodo previo registrado y proseguir con un modelo de captura de datos continuo.

En RGBDSLAM se sigue el segundo procedimiento, que aunque implica una existencia de error mayor, facilita el proceso de comparación y asociación general.

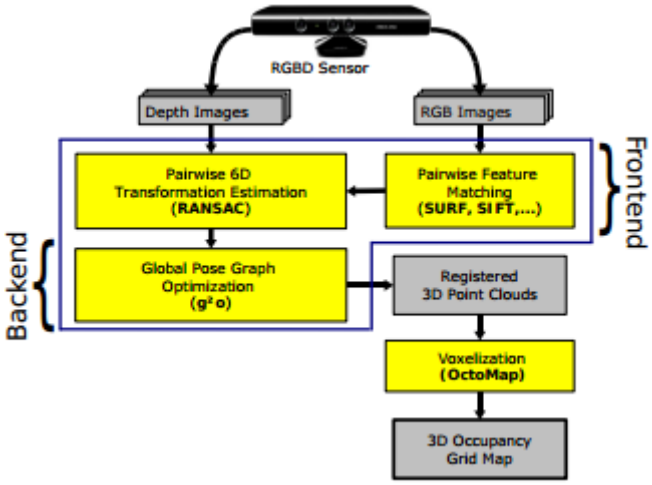



Ilustración 14 - Diagrama de funcionamiento del software RGBDSLAM

### 3.3.2.2 Backend

El movimiento propio de la cámara se analiza tomando como datos la posición propia asociada a cada nodo incluido en el proceso SLAM.

Por medio del método g2o, se optimiza el gráfico obtenido de la variación de la posición durante el registro de los distintos nodos, permitiéndonos obtener una trayectoria consistente. Una optimización en la trayectoria permite detectar más fácilmente el punto de cierre del bucle.

### 3.3.2.3 Mapeado

Utilizando los resultados tanto de la captura de datos en la sección Frontend y el procesamiento del desplazamiento y la posición de la cámara, es posible pasar a la generación del mapa tridimensional.

En este paso entra otra vez en juego la búsqueda de reducir la carga computacional del proceso global, por lo que en vez de recurrir a la representación punto a punto de los datos obtenidos por el sensor, se procede a aplicar Octomap como forma conseguir una representación fiel del entorno estudiado a la vez que se reduce el tiempo y la consumición de recursos para llevarlo a cabo.

La conclusión que se obtiene del resumen sobre el funcionamiento del software, es la disponibilidad de una herramienta que busca la funcionalidad de la obtención de datos antes que una precisión exhaustiva en los mismos.

El software en su versión con interfaz gráfica es muy intuitivo y permite al usuario comenzar con le adquisición de nubes de puntos de forma inmediata. Hay muchísimos

parámetros a modificar según queramos ajustar los requerimientos del programa a máquinas más o menos potentes, o a un detalle y número de puntos mayor o menor. Estos parámetros no pueden alterarse desde el propio menú de la interfaz grafica, por lo que se deben tocar los archivos de código originales. Por suerte, estos vienen abundantemente documentados, proporcionando indicativos del efecto que puede tener alterar cada uno de los parámetros del programa.

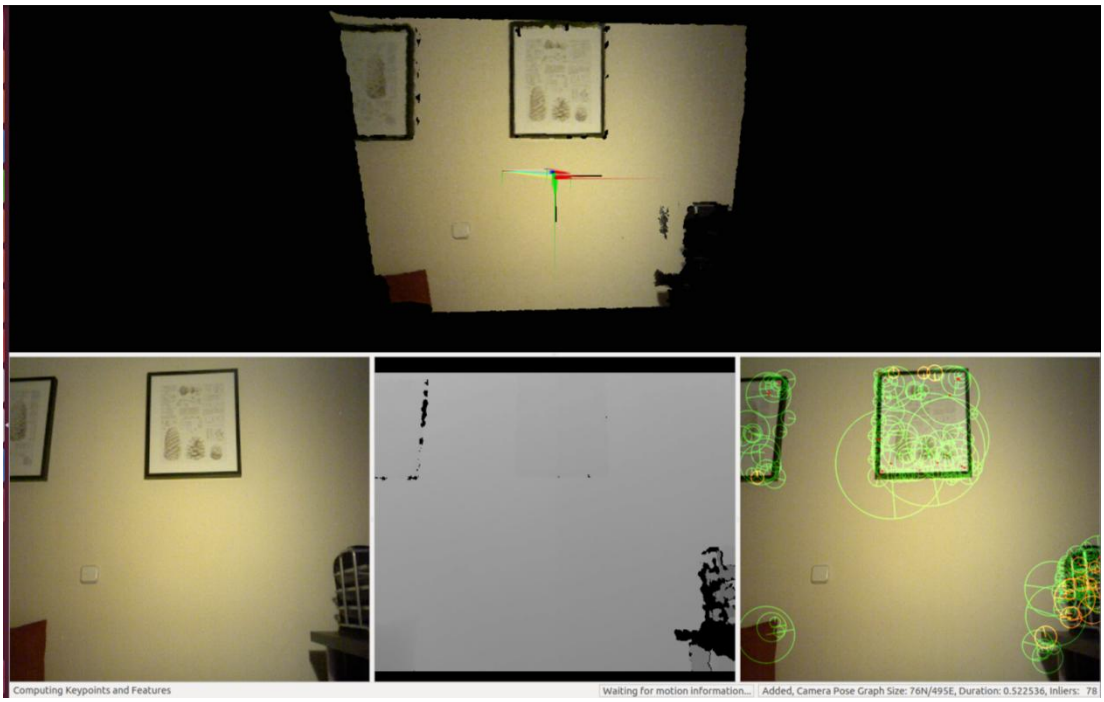



Ilustración 15 – Vista general de la interfaz gráfica de RGBDSLAM en modo de procesamiento

**Imagen inferior izquierda**

Se trata de la ventana de visualización de los datos RGB.

**Imagen inferior central**

Datos de profundidad capturados por la cámara.

**Imagen inferior derecha**

Aplicación visual del algoritmo de detección (se puede observar con mayor claridad en movimiento)

**Imagen superior**

Representación tridimensional del mapa. Se pueden observar unos ejes de referencia situados en el centro de la imagen. Corresponden a la posición de la cámara en cada uno de los instantes. Este mapa se puede rotar y trasladar.

### 3.3.2.4 *Parámetros de algoritmos*

**Tabla 3- Parámetros relacionados con los algoritmos del software RGBDSLAM**

| Nombre | Valor | Información |
|---|---|---|
| store_pointclouds | TRUE/FALSE | True: Si se quiere obtener la trayectoria<br>Falso: Reduce el consumo de memoria |
| min_translation_meter | Númerico | Valor de movimiento mínimo para capturar una nueva instantánea |
| min_rotation_degree | Númerico | Valor de rotación mínimo para capturar una nueva instantánea |
| visualization_skip_step | Númerico | Registra una de cada N entradas en el archivo .pcd (donde N es el número introducido en el parámetro) |
| visualize_keyframes_only | TRUE/FALSE | Visualización de ciertos fotogramas |
| min_time_reported | Númerico | Mantener en torno a 0,01 para optimizar el análisis en tiempo real |
| octomap_online_creation | TRUE/FALSE | Valida o no la creación del mapa Octomap |
| octomap_resolution | Númerico | Define la resolución del mapa |
| use_icp | TRUE/FALSE | Uso del algoritmo ICP |
| icp_method | TRUE/FALSE | Depende de use_icp |
| gicp_max_cloud_size | Númerico | Tamaño máximo de la nube de puntos para el cálculo |
| ransac_iterations | Númerico | Número de iteraciones RANSAC |
| g2o_transformation_refinement | Númerico | Nivel de optimización de g2o |
| predecessor_candidates | Númerico | Número de fotogramas precedentes candidatos |
| neighbor_candidates | Númerico | Selección de candidatos próximos |
| min_sampled_candidates | Númerico | Número mínimo de fotogramas candidatos a considerar |
| observability_threshold | Númerico | Umbral de observación |

### 3.3.2.5 *Parámetros de datos de entrada*

**Tabla 4 - Parámetros asociados con la captura de datos en el software RGBDSLAM**

| **Nombre** | **Valor** | **Información** |
|---|---|---|
| feature_detector_type | Predefinido | Tipo de detector de características |
| feature_extractor_type | Predefinido | Tipo de extractor de características |
| nn_distance_ratio | Númerico | |
| max_keypoints | Númerico | Número máximo de puntos clave |
| min_matches | Númerico | Número mínimo de coincidencias |

#### *3.3.2.6 Modo de funcionamiento*

Una vez situada la cámara en la posición inicial deseada, se comienza a ejecutar el programa.

El usuario debe realizar el movimiento de captura de la cámara conforme a la exactitud deseada en la obtención de datos. Hay que tener en cuenta que una mayor precisión implica unas Point Clouds cada vez más grandes, que pueden dificultar el proceso de grabado.

El software cuenta con diversos modos de lanzamiento o "launchers". Cada launcher lleva asociado una funcionalidad o serie de características y el usuario debe decidir cual quiere utilizar. Así:

*roslaunch rgbdslam rgbdslam.launch*

Abriría la versión más básica del programa.

Por otro lado, se ha comparada el funcionamiento en el portátil utilizado de las dos siguiente opciones de software:

*roslaunch rgbdslam kinect+rgbdslam.launch*

*roslaunch rgbdslam slow_computer.launch*

Al ser un portátil de no mucha potencia de procesamiento, las opciones variaban entre utilizar kinect+rgbdslam (versión básica con unos parámetros ajustado para un procesador con características medias-altas) y retocar estos parámetros disminuyendo la consumición de memoria, o bien utilizar la versión predefinida para procesadores de poca potencia, pero ajustando los parámetros de manera que estos ofreciesen una mayor eficiencia a la tarea de realizar el proceso SLAM.

Finalmente se ha empleado esta segunda opción, y los resultados que defienden esta toma de decisión se proporcionan en un capítulo posterior del proyecto. En general se puede concluir que la proporción de las nube de puntos que pueden ser capturadas es mayor (sacrificando el detalle de las mismas), y por lo tanto se puede obtener un modelo más representativo del entorno.

Una de las diferencias más importantes entre los dos "launchers" es el modo de captura de datos visuales, basándose la opción por defecto en SURF y la otra opción diseñada para mejorar el rendimiento en máquinas de poca potencia en ORB.

### 3.3.2.7 Opciones del software

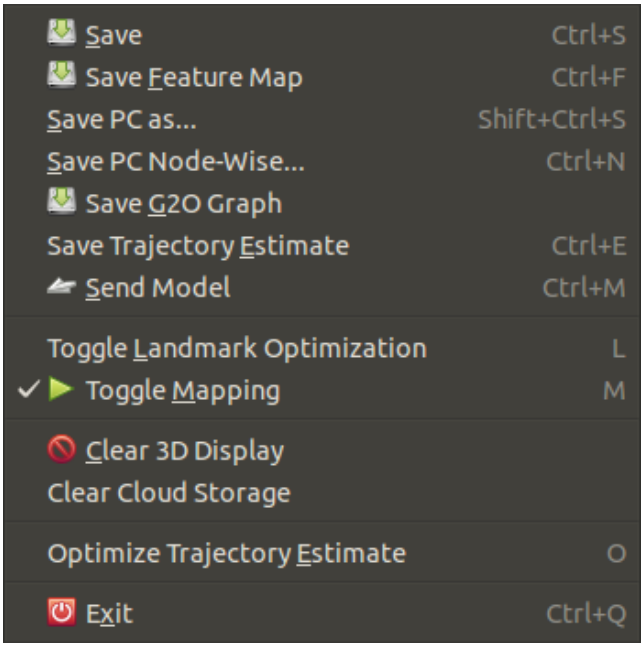



Ilustración 16 – Menú Data

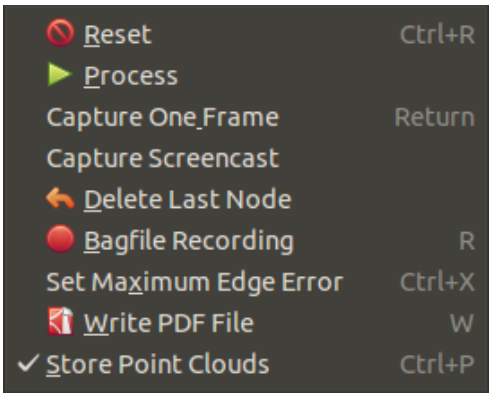



Ilustración 17 – Menú Processing

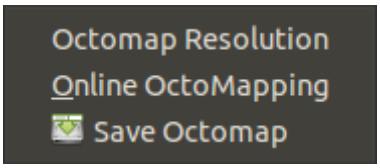



Ilustración 18 – Menú Octomap

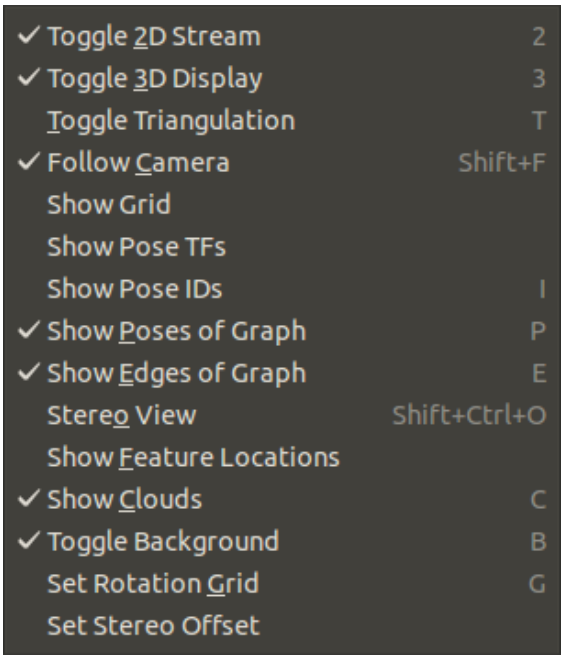



Ilustración 19 – Menú View

Tabla 5 - Opciones del software RGBDSLAM

| Opción | Funcionalidad |
|---|---|
| Save Feature Map | Guardar la representación de los principales puntos de interés del entorno capturado |
| Save PC as | Guardar la nube de puntos global. |
| Save PC Node-Wise | Guardar en formato .pcd los distintos fotogramas asociados a cada nodo |
| Save G2O Graph | Guadar el grafo optimizado por el “framework” g2o |
| Save Trajectory Estimate | Guardar en archivo .txt las variables asociadas a la trayectoria estimada seguida por la cámara durante el proceso de captura. |

### *3.3.2.8 Algoritmos y frameworks*

**ICP**

El Iterative Closest Point algorithm se utiliza para obtener la correspondencia entre dos nubes de puntos tomadas desde distintos puntos en el espacio.

Parámetros de entrada:

- Point Cloud actual
- Point Cloud de referencia

Parámetros de salida:

- Matriz de transformación entre ambas Point Clouds

El funcionamiento básico consiste en realizar una comparación y asociación de ambas nubes de puntos identificando los puntos más cercanos entre sí en ambas escenas.

Una vez obtenidas estas distancias mínimas, el algoritmo prioriza una serie de pares de puntos, desechando a su vez pares de puntos que no superan el umbral de error. Se repiten los pasos de comparación y asociación hasta encontrar una convergencia aceptable.

El ICP se caracteriza por su convergencia a mínimos locales, con lo que una buena inicialización del algoritmo es vital para su correcto funcionamiento. Considerando la gran cantidad de puntos con los que cuentan nuestros modelos 3D, la aplicación del ICP en la totalidad de la nube de puntos es inviable, y es por lo que se recurre a realizar la operación sobre una muestra representativa de los mismos.

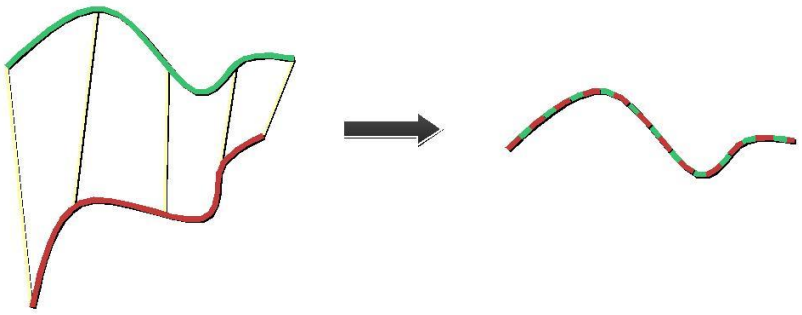


Ilustración 20 – Concepto del algoritmo ICP

Este algoritmo no se utiliza expresamente en el Software, pero representa una alternativa a la aplicación de RANSAC.

## RANSAC

RANSAC consiste en un proceso iterativo para encontrar el modelo al que más se ajusten los datos que se están estudiando. Mediante una solución inicial y un conjunto dado de iteraciones, se pueden encontrar los parámetros del modelo dado un conjunto con una presencia alta de valores atípicos

A la hora de analizar este método, hay que definir dos tipos de datos en el modelo inicial:

- Inliers: se trata de los valores que nos proporcionarían la solución óptima al problema.
- Outliers: estos valores atípicos son resultado directo de posibles errores o una alta variabilidad en los datos obtenidos.

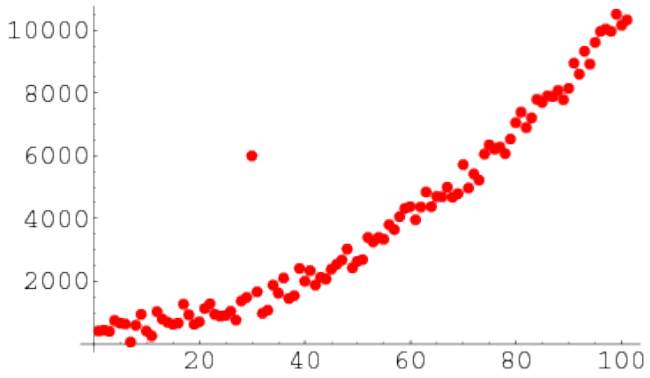



Ilustración 21 – Ejemplo de outlier en un conjunto de datos

La aplicación del algoritmo RANSAC sigue los siguientes pasos:

1. Se seleccionan los parámetros iniciales
    a. Número mínimo de puntos que proporcionan una solución satisfactoria (se establece un porcentaje del número total de datos iniciales del modelo)
    b. Número de iteraciones a realizar
2. Se escoge una solución inicial aleatoria y se comprueba su validez de cara al parámetro seleccionado en el punto 1. En caso de superar este umbral, se finaliza la ejecución del algoritmo.

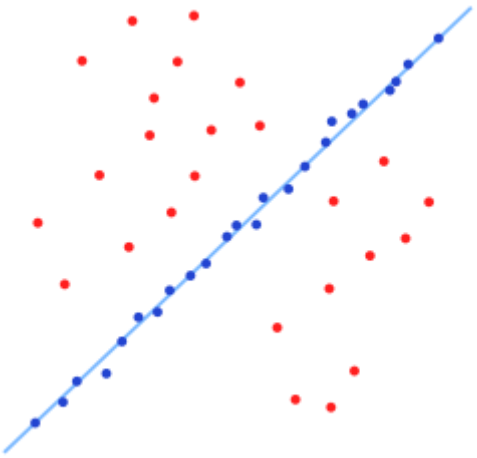


Ilustración 22 – Solución que aglutina el mayor número de inliers

3. En caso de no haber llegado a una solución en el apartado anterior, iterar un número N de veces.

Ventajas:

- Solución inicial aleatoria, no hacen falta muchos datos.

Desventajas:

- No se garantiza la obtención de la solución óptima, el algoritmo se detiene al encontrar la primera solución que el usuario ha definido como válida.

## g$^2$o

El "General Graph Optimization" supone un conjunto de técnicas (no es exactamente un algoritmo, sino un "framework") utilizadas para la optimización y corrección de errores en problemas que se pueden modelar por medio de gráficos.

Como se ha especificado anteriormente, el problema SLAM se puede considerar desde una perspectiva en dos dimensiones, pudiendo representar tanto las posiciones del robot como los "landmarks" identificados en el entorno dentro en un gráfico 2D.

El parámetro de entrada para aplicar este método es la función de error del sistema con el que vamos a trabajar. En este caso aplicamos g2o para la estimación de la trayectoria de la cámara.

La trayectoria de la cámara viene proporcionada visualmente por RGBDSLAM e incluida en el modelo 3D generado por Octomap.

## Octomap

Se trata de otro "framework" destinado a la representación tridimensional de modelos. A la hora de representar un modelo en volumen hay que tener en cuenta varios conceptos al aplicar Octomap:

- Representación probabilística

  En todas las aplicaciones robóticas asociadas a mediciones de datos con sensores de distancia hay presente una incertidumbre (un problema mayor en el caso de este proyecto debido a la poca especialización de la cámara utilizada). Es por lo tanto que hay una probabilidad asociada a dicha incertidumbre a la hora de representar el modelo 3D.

- Octrees

  El "octree" constituye una estructura de datos que tiene como objetivo dividir y segmentar un modelo 3D. Cada uno de los octantes se subdivide a su vez, avanzando cada vez un nivel de profundidad siendo este proporcional a la precisión deseada.

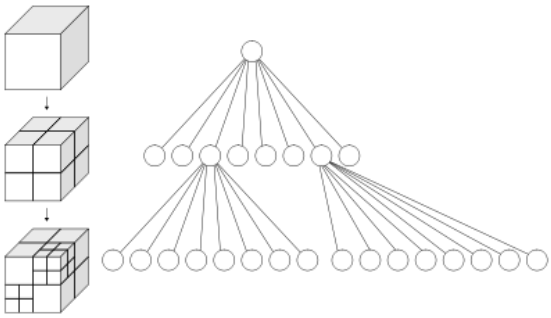


Ilustración 23 – Concepto de crecimiento exponencial

Ventajas de Octomap:

- Alta compresibilidad de memoria.
- Capacidad de representar modelos 3D con componentes de color.
- Representación de modelos con zonas desconocidas o vacías.

### 3.3.3 Detectores de puntos de interés

La detección de puntos de interés es una operación en el tratamiento de imágenes que como su propio nombre indica consiste en la selección e identificación de características representativas y destacables dentro de la información proporcionada por una imagen o fotograma.

Introducido el concepto de punto de referencia o “landmark”, los “feature detectors” constituyen el primer paso para extraer estas referencias dentro del modelo.

La detección puede dividirse en varios tipos dependiendo de la característica que en el problema se ha definido como objetivo de la identificación:

- Bordes
- Esquinas
- Cúmulo
- Arrugas

Estas características no son exclusivas de un detector a otro y pueden ser combinadas dependiendo de la aplicación estudiada.

En el software, somos capaces de observar los elementos a los que el detector de puntos de interés localiza en cada momento:

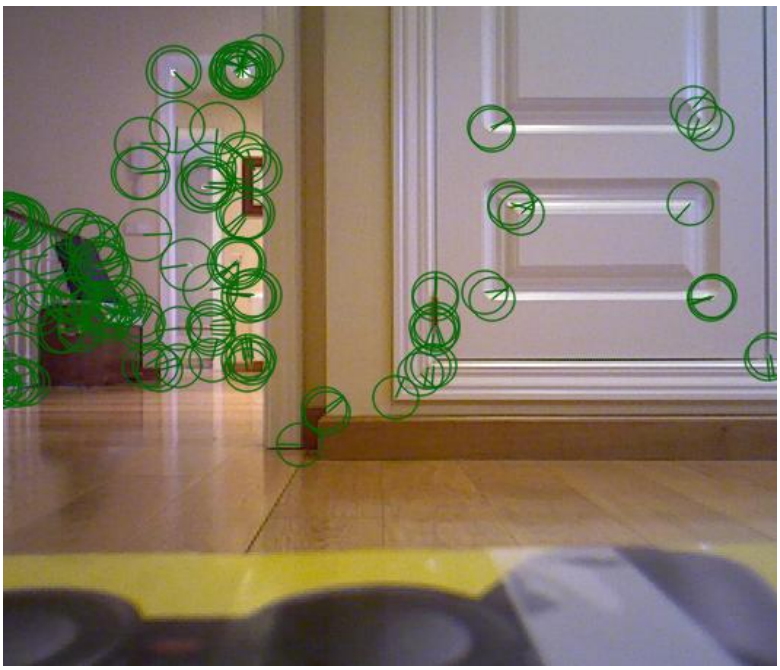


Ilustración 24 - Detección de puntos de interés en la ventana RGBDSLAM

#### 3.3.3.1 Datos obtenidos por el programa

Del software obtenemos los siguiente datos:

a) Archivo .pcd
b) Archivo .pcd (nodewise)
c) Archivo estimación de trayectoria .txt
d) Archivo .ply
e) Archivo g2o

#### 3.3.3.2 Características de una Point Cloud

Una Nube de Puntos es un archivo guardado con extensión .pcd (point cloud data), que dependiendo del hardware con el que se han obtenido los datos nos proporcionará una u otra información.

Generalmente estos archivos cuentan con un encabezado que proporciona información básica de la estructura de datos del archivo .pcd, como pueden ser el código (ASCII, etc) en el que se dan los datos, la existencia o no de las componentes de color, etc.

A continuación, se proporcionan las coordenadas de cada puntos recogido en xyz, así como la componente de color R, G o B dada en formato decimal pero correspondiendo a un número de 8-bits. Teniendo en cuenta el número de puntos que se recogen en cada muestra, al final se obtienen archivos que dependiendo de los parámetros fijados en el programa varían de un rango de 250 Mb hasta 1Gb de información.

Características:

- Versión
- Campos
    - Coordenadas
    - Coordenadas + Componentes de color
    - Coordenadas + Vectores normales a la superficie
    - Etc
- Tamaño
- Tipo de codificación
    - Con signo
    - Sin signo
    - Floats
- Ancho
- Alto
- Puntos
- Datos

Tabla 6 - Formatos PCD y PLY

**PCD Format**

```
# .PCD v.7 - Point Cloud Data file format
VERSION .7
FIELDS x y z rgb
SIZE 4 4 4 4
TYPE F F F F
COUNT 1 1 1 1
WIDTH 213
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 213
DATA ascii
0.93773 0.33763 0 4.2108e+06
0.90805 0.35641 0 4.2108e+06
0.81915 0.32 0 4.2108e+06
0.97192 0.278 0 4.2108e+06
0.944 0.29474 0 4.2108e+06
0.98111 0.24247 0 4.2108e+06
0.93655 0.26143 0 4.2108e+06
0.91631 0.27442 0 4.2108e+06
0.81921 0.29315 0 4.2108e+06
0.90701 0.24109 0 4.2108e+06
0.83239 0.23398 0 4.2108e+06
0.99185 0.2116 0 4.2108e+06
```

**PLY Format**

```
ply
format ascii 1.0
element vertex 16823635
property float x
property float y
property float z
property uchar red
property uchar green
property uchar blue
end_header
1.23503   0.666092   0.508333   25  25  25
1.23504   0.661384   0.508333   26  25  32
1.23104   0.654421   0.506688   26  26  26
1.24004   0.654766   0.510389   27  27  27
1.23505   0.647258   0.508333   27  27  27
1.23505    0.64255   0.508333   30  27  19
1.23106   0.635647   0.506688   27  27  27
1.23106   0.630954   0.506688   30  27  19
1.22607   0.623557   0.504631   30  25  27
1.22208   0.616734   0.502986   26  25  32
1.21808   0.609943   0.501341   31  29  43
1.21809   0.605299   0.501341   29  29  29
1.21809   0.600655   0.501341   29  29  29
```

En esta tabla se pueden observar los 2 formatos más utilizados para la manipulación y visualización de Point Clouds.

Además de la extensión .pcd, encontramos otro tipo de archivo con extensión .ply (polygon format), que cuenta con la ventaja de ser compatible con un número mayor de programas como Mathematica y MeshLab. El Polygon Format guarda la información en forma de planos poligonales, siendo capaz de almacenar información relativa a la transparencia y la textura de los datos tridimensionales escaneados.

Como elemento de utilidad en este proyecto, el polygon format permite la visualización de nubes de puntos de forma más intuitiva.

#### *3.3.3.3 Tratamiento de la Nube de puntos*

Una vez obtenida la Point Cloud, se deben aplicar una serie de pasos para que la estructura de puntos resultante sea la adecuada para trabajar con ella.

**Filtrado**

Para comenzar, hay que saber que las Point Clouds obtenidas con el software contienen varias decenas de millones de puntos, por lo que la tarea de trabajar con estructuras tan grandes y con tanta información se hace inviable. Es por esto que optamos por aplicar un filtro simple a la Point Cloud, con el fin de reducir el tamaño de archivos .pcd con el que vamos a trabajar y para facilitar la velocidad de procesamiento para las distintas aplicaciones de los mapas tridimensionales obtenidos. Mediante ese filtro simple logramos obtener una reducción de la densidad de puntos que nos proporciona archivos con un 5-10% (depende de la Point Cloud filtrada en cada caso, ya que según la naturaleza de la misma habrá zonas donde puedan eliminarse mayor número de puntos sin perder nada de información, como pueden ser paredes, techo, etc) y sin llegar a perder una parte significativa de la información que consideramos útil para el proyecto.

Cabe destacar que el objetivo del proyecto no es tratar con las zonas detalladas de la habitación o conjunto de habitaciones en cuestión, sino de trabajar con un modelo simplificado del contorno general de dicha habitación, por lo que para el filtro diseñado para estas Point Clouds, no debemos preocuparnos por la posible pérdida de detalle en la simplificación de la nube de puntos como se ha mencionado anteriormente.

El filtro aplicado conserva el archivo original a la vez que genera el archivo propiamente filtrado, dando la posibilidad al usuario de tener siempre disponible el mapa 3d original para poder utilizarlo en aplicaciones con diferentes utilidades.

El archivo filtrado, al cual por defecto se le añade la extensión _filtered para su mejor identificación, es con el que se trabaja para las tareas de extracción de planos y contornos, lo que agiliza enormemente todas las operaciones de procesamiento posteriores, al tener que trabajar con un número de puntos mucho menor.

Este filtrado simple consiste en un “downsampling” de la nube de puntos original. El concepto de “voxel_grid” es ya representativo de la técnica que se va a utilizar.

Con un concepto parecido al de los Octrees mencionados a la hora de aplicar Octomap, la técnica de voxel_grid consiste en generar un mallado tridimensional del entorno, con unas dimensiones definidas por el usuario como parámetros en el programa.

Seguidamente, todos los puntos contenidos en dicho cubo (la forma más sencilla de aplicar el mallado) son llevados al centroide del mismo, condensando la información

de un número indeterminado de puntos (este número dependerá de la densidad de información en esa región de la nube de puntos) en uno solo.

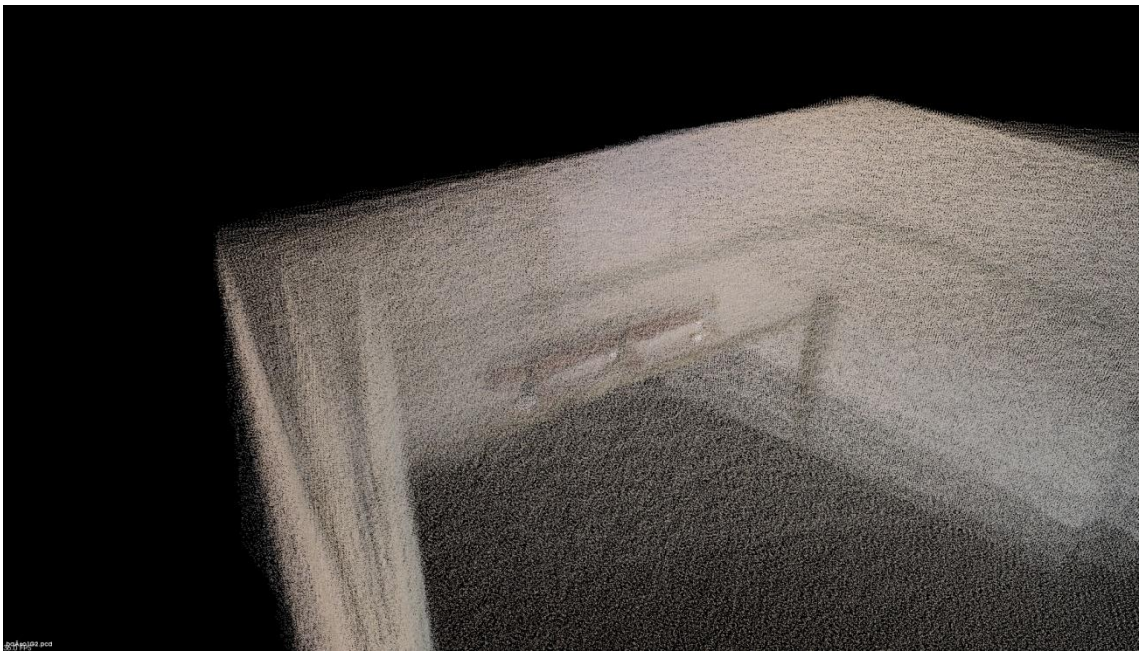


Ilustración 25 – Point Cloud pre-filtrado

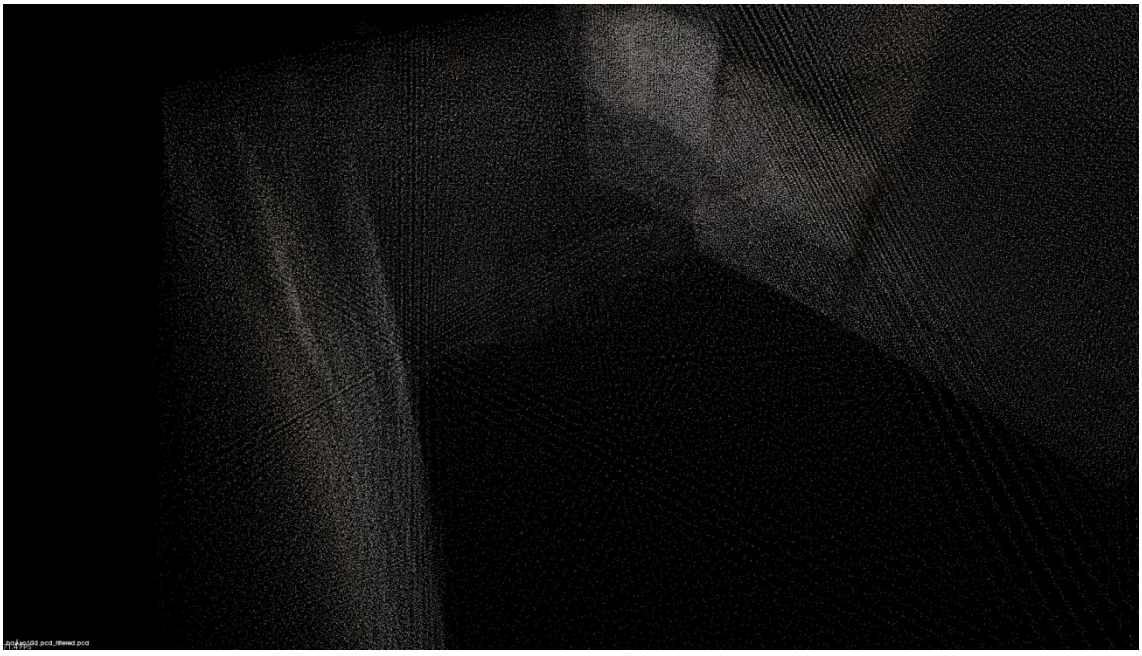


Ilustración 26 – Point Cloud post-filtrado

**Suavizado**

Uno de los problemas que tiene el software RGBDSLAM, y el cual no puede evitarse, son los errores en tiempo de captura de datos. Ciertos errores son inevitables pero si se pueden corregir a efectos de la visualización, y es por esto que se aplica una etapa de suavizado de superficies en busca de proporcionar una continuidad a las superficies de contorno. Esta continuidad no siempre se consigue de la forma deseada a la hora de capturar los datos con la cámara ASUS, por lo que mediante la definición de ciertos parámetros dentro del pequeño programa de suavizado, podemos elegir el grado de offset que se va a aceptar como válido a la hora de crear una continuidad en la superficie para evitar los saltos que se producen en la grabación.

Por defecto hay un valor moderado seleccionado. Es necesario saber que cuanto mayor es el rango que aplicamos para definir la continuidad de la superficie, mucho mayor es el tiempo de procesamiento que conlleva suavizar las superficies de la nube de puntos.

Este paso es en algunos casos opcional, ya que las partes posteriores del programa no trabajan necesariamente con la continuidad de una superficie, ya que esta continuidad puede no constituir un plano. Por lo tanto, es una parte del programa destinada principalmente al tratamiento para la visualización de las Point Clouds obtenidas, y no tanto para su manipulación de cara a los objetivos que se quieren conseguir en el trabajo.

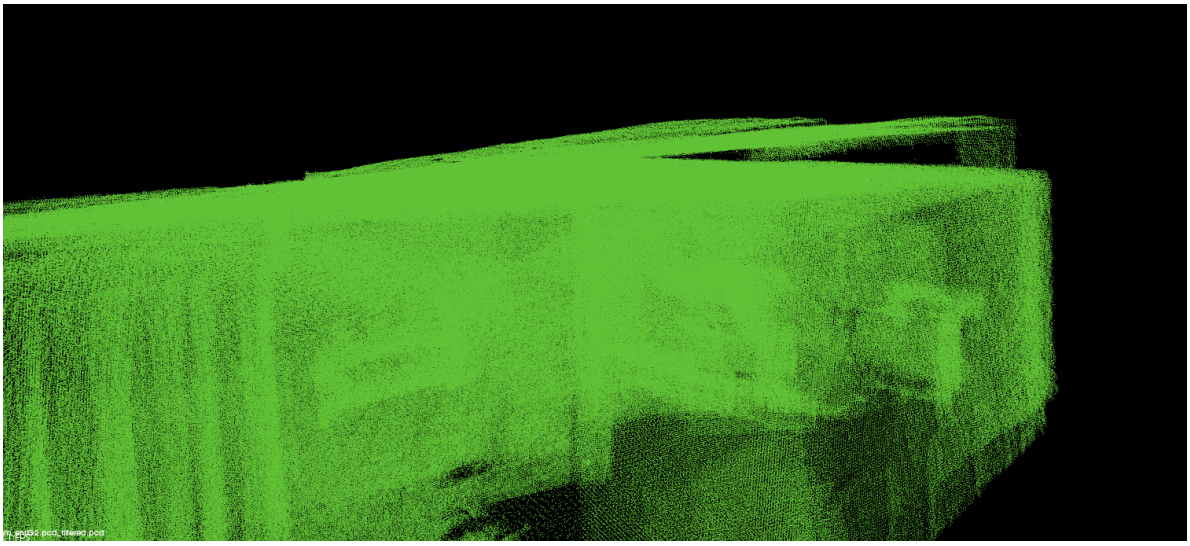


Ilustración 27 - Error de discontinuidad en los datos

**Extracción de planos**

Es esta parte del proceso de tratamiento de las Point Clouds aplicaremos una segmentación al conjunto filtrado con el fin de aislar los planos que componen el entorno capturado. En un principio el interés iba a estar centrado tanto en el techo como en el suelo de la estancia considerada en cada caso. Con la decisión de optimizar tiempo de cálculo y simplificar los modeles, con el techo es suficiente en estos casos.

La extracción de planos es un paso crítico a la hora de obtener la representación final. El acarreo de errores en el tiempo de captura de los entornos puede llevar a posibles obtenciones de planos de una superficie que si bien en la realidad aparece como continua, se vea segmentada por el propio algoritmo utilizado al hallar este una variación importante en la continuidad de la misma.

Con este problema en mente, se pretende aplicar al algoritmo utilizado un rango variable de severidad a la hora de interpretar la continuidad de los planos.

**Extracción del contorno**

Este último paso del tratamiento de la nube es el que nos proporcionará la representación de cada plano simplificado en dos dimensiones para ser capaces de modelar en entorno global de la planta en el mapa final. Depende directamente del proceso de extracción de planos, ya que esta parte del tratamiento solo va a actuar sobre dichos planos, y no sobre la nube de puntos total.

### 3.3.4 Point Cloud Library

Point Cloud Library no constituye un software o programa propiamente dicho. Se trata de un conjunto de librerías diseñadas para el manejo de nubes de puntos. Si bien las posibilidades son enormes, la extensa lista de tutoriales y programas básicos proporcionados constituyen un conjunto de herramientas suficiente para realizar la mayoría de procesos de tratamiento de nubes de puntos que se requieren para la realización del objetivo de este proyecto.

A su vez, la versión instalada de ROS incluye por defecto un conjunto de librerías de ROS, aunque algunas versiones anterior a la que se puede obtener de la página oficial. Por este motivo, y al disponer de 2 ordenadores para realizar el proyecto, se ha decidido probar los dos métodos, dejando la versión más actual para las tareas de procesamiento más complejas y la más antigua para la visualización más básica de las nubes de puntos.

Las principales áreas de aplicación de las librerías incluidas en el proyecto Point Cloud Library son:

- Filtros
- Características
- Puntos clave
- Captura
- Kdtree
- Octree
- Segmentación
- Sample Consensus
- Superficies
- Identificación
- Lectura y Escritura
- Visualización

### 3.3.5 Cloud Compare

Este software, al contrario que los demás empleados en el proyecto, ha sido utilizado en Windows. Se trata de una herramienta de visualización y manipulación de nubes de puntos. De cara a este trabajo, presenta una serie de funcionalidades muy útiles como:

- Visualización de la nube de puntos con las componentes de color
- Alineación de varias nubes de puntos
- Rotación y traslación de nubes de puntos

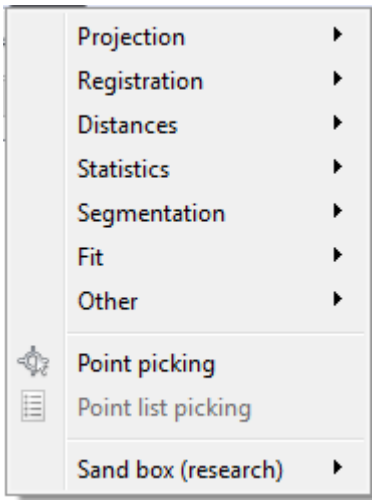



Ilustración 28 - Menú de opciones del software CloudCompare

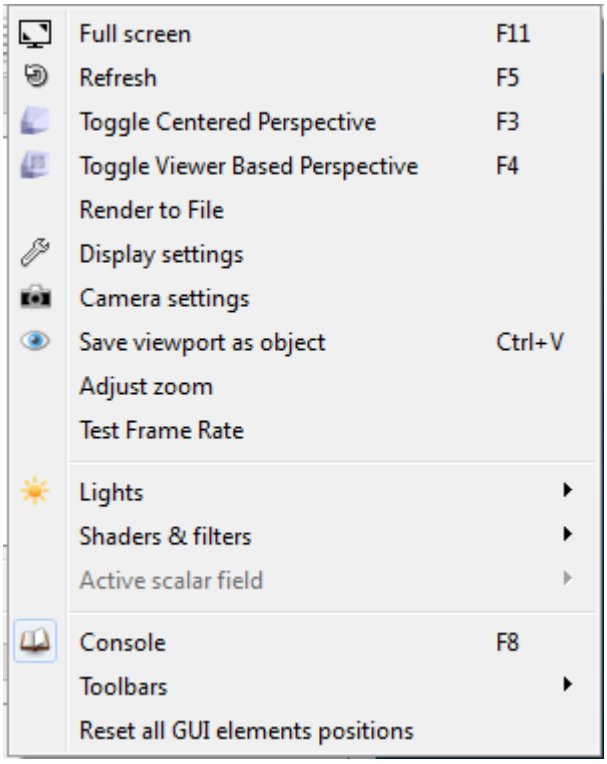



Ilustración 29 - Menú de visualización del software CloudCompare

La finalidad de este software es combinar todos los contornos de Point Clouds que se obtendrán eventualmente y construir el plano de la planta de todo el entorno utilizado.

Con las herramientas de alineación y desplazamiento se pretende lograr que los errores posibles errores (de desplazamiento, pérdida de información) no impidan obtener una vista final del conjunto, la cual sea de utilidad para la aplicación deseada.

## 3.4 Planteamiento del problema

### 3.4.1 Métodos para la obtención del mapa global

#### *3.4.1.1 Solución local o modular*

Para este proyecto, se van a aplicar dos metodologías distintas a la hora de intentar reproducir una representación adecuada del entorno estudiado.

Se ha presentado ya la opción de realizar la captura de forma modular, es decir, obtener un mapa local para cada estancia constituyente de la planta entera y aplicar luego una fusión de estos con el fin de obtener el mapa global.

Las ventajas de este método están constituidas por la capacidad de dotar a la representación individual de cada estancia un mayor detalle y una serie de procesos de acondicionamiento exclusivos para cada una. Es un método apropiado para este tipo de estudios, y especialmente útil a la hora de gestionar la información de la que va a disponer el robot al desplazarse por el entorno en aplicaciones posteriores. El proceso que seguiría un supuesto robot utilizando el mapa generado por un robot aplicando SLAM sería el siguiente:

1. Utilización del mapa global para obtener una referencia constante de la posición propia con respecto al entorno.
2. Navegación por este mapa global, al estar el mapa subdividido en estancias (los módulos del mapa), el robot tiene presente en cada momento en cuál de estos módulos se encuentra.
3. El robot, con esta información, es capaz de acceder al mapa 3D completo del módulo (éste podría ser la representación tridimensional filtrada de cada estancia, para reducir la memoria utilizada). La ventaja de este método radica en la manipulación una a una de las estancias, lo que dota al robot con la capacidad para trabajar de forma cómoda con el mapa tridimensional detallado y poder sortear y visualizar eventuales obstáculos en el entorno. Si la aplicación del robot estuviese condicionada por la necesidad de obtener un trayecto libre de obstáculo, el tener que acceder a un mapa global de toda la planta requeriría mucha más potencia de procesamiento, a la vez que se perdería un nivel de detalle que el método modular siempre nos proporciona.

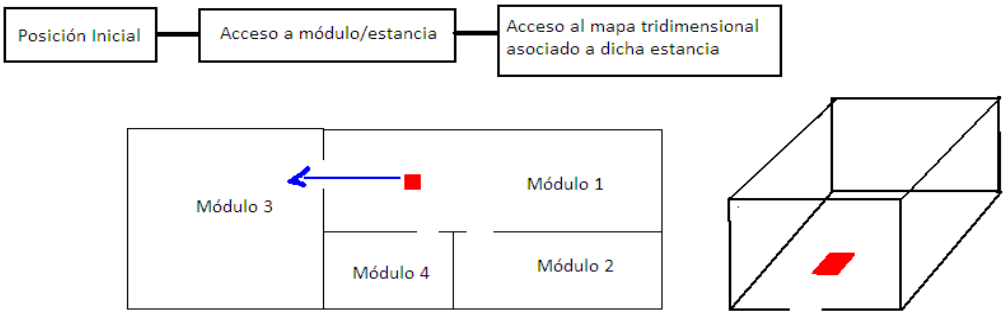



Ilustración 30 - Simplificación del problema modular

Para la aplicación de este método, la representación de la trayectoria (que es la que el robot debe cotejar para interpretar su posición) debe estar también segmentada de manera que se corresponda con la división en módulos del entorno realizada.

Sin embargo, al haber obtenido mapas de trayectoria individuales para cada estancia, no es posible realizar la correlación con ningún mapa global de trayectoria obtenido por SLAM. Este mapa global de trayectorias debe de proporcionarse de manera externa al proceso de localización y mapeado realizado.

La fuente de error más evidente surge al moverse el robot por las zonas próximas a los cambios entre módulos.

#### *3.4.1.2 Solución global*

Esta solución es la que se emplea normalmente al realizar experimentación mediante SLAM.

Se trata de realizar el proceso de forma continua y cubriendo la totalidad del entorno a estudiar, generando un mapa topogeométrico global.

Para este caso, el robot sigue una trayectoria, significativamente más extensa que en la solución previa, que permite acceder a todas las regiones del entorno.

Una vez realizado dicho recorrido, el usuario dispone de la siguiente información:

1. Grafo de la trayectoria global del robot, con los nodos representados.
2. Mapa tridimensional parcial del entorno. Hay que tener en cuenta que para este método la finalidad del SLAM es obtener un mapeado unificado del entorno. Si tenemos en cuenta el alcance del proyecto, combinar la búsqueda de detalle con esta solución se hace inviable debido a la carga computacional que supondría gestionar el mapa completo. Por lo tanto obtenemos un único mapa con un contenido más “esquemático” del entorno, pero más práctico a efectos de navegabilidad.

En este caso, sé que tenemos un mapa de trayectoria global, siendo capaz del robot de acceder a los fotogramas asociados a cada nodo si fuese necesario.

Al final, utilizando ambas soluciones, un supuesto robot que haga uso de cualquiera de las opciones de mapas generados, va a tener presente un conjunto de zonas “oscuras” (correspondientes a los contornos de las paredes, y puede que en el caso de la solución modular también a los obstáculos individuales de cada habitación) que le van a permitir la navegación por el entorno de una forma razonablemente segura.

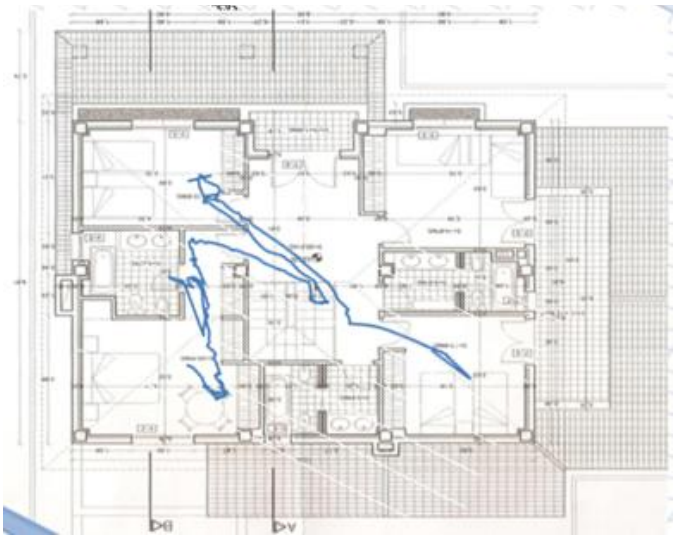


Ilustración 31 - Trayectoria a lo largo de varias estancias

De manera aproximada se ha superpuesto la trayectoria estimada al mapa real del entorno para uno de los experimentos realizados. Como se puede observar, se distinguen zonas donde el error acumulado durante el recorrido sitúa al robot simulado en zonas inaccesibles.

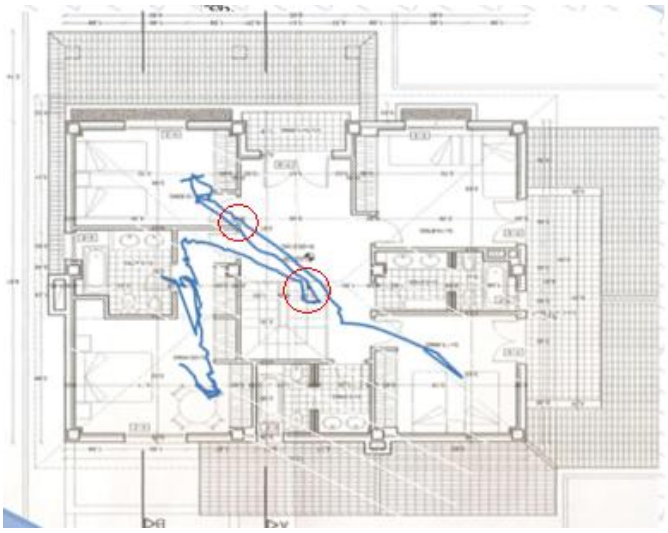


Ilustración 32 - Regiones en las que se obtiene error de la trayectoria estimada

Tabla 7 - Extracto de coordenadas asociadas a varios nodos en orden temporal

| Referencia Temporal | Coordenada X | Coordenada Y | Coordenada Z | q1 | q2 | q3 | q4 |
|---|---|---|---|---|---|---|---|
| 1410885163,39747 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1410885165,84647 | 0,014452 | 0,011204 | -0,009426 | -0,009841 | -0,001951 | -0,028786 | 0,999535 |
| 1410885166,28714 | 0,030766 | 0,010966 | -0,008489 | -0,013022 | -0,001149 | -0,057496 | 0,99826 |
| 1410885166,61915 | 0,047041 | 0,014403 | -0,001822 | -0,01354 | 0,001767 | -0,088331 | 0,995998 |
| 1410885167,01792 | 0,061857 | 0,008485 | -0,004308 | -0,017684 | 0,001627 | -0,119731 | 0,992648 |
| 1410885167,21792 | 0,073369 | 0,009957 | -0,011799 | -0,024703 | -0,000384 | -0,14796 | 0,988685 |
| 1410885167,48576 | 0,09365 | 0,011697 | -0,023509 | -0,030084 | -0,004818 | -0,187098 | 0,981869 |
| 1410885167,68620 | 0,102787 | 0,00781 | -0,017435 | -0,032086 | -0,000899 | -0,204687 | 0,978301 |
| 1410885168,02142 | 0,118664 | 0,005843 | -0,023006 | -0,034051 | -0,003048 | -0,227956 | 0,973071 |
| 1410885168,28943 | 0,143573 | -0,002198 | -0,024977 | -0,038257 | -0,00226 | -0,260201 | 0,964794 |
| 1410885168,52546 | 0,172348 | -0,014525 | -0,030833 | -0,038401 | -0,004016 | -0,28517 | 0,957699 |
| 1410885169,03099 | 0,198735 | -0,032129 | -0,027138 | -0,043471 | 0,001179 | -0,311756 | 0,949167 |

La trayectoria representada ha sido obtenida mediante un gráfico en dos dimensiones con las coordenadas X e Y de la estimación de la trayectoria proporcionada por el software (ver Tabla situada más arriba). A continuación se adjunta una trayectoria teórica de las condiciones en las que se realizaron las mediciones (aproximada con rectas en los trayectos lineales).

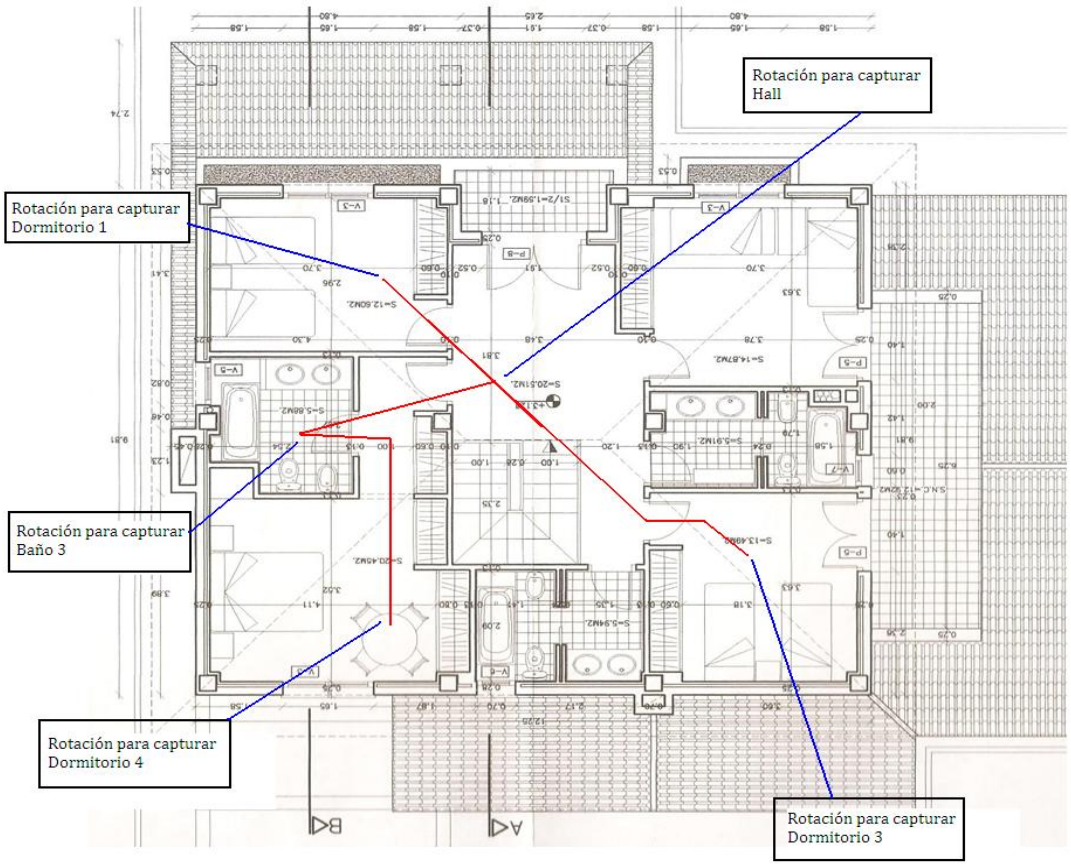



Ilustración 33- Trayectoria teórica

# 4. Tratamiento de Point Clouds

Sin embargo, hay que hacer un análisis de los tratamientos mencionados y la composición y aspecto de las Point Clouds durante los pasos intermedios del proceso.

Para la visualización de los distintos resultados y ejemplos de las Point Clouds empleadas, se utilizar una herramienta de la Point Cloud Library de la biblioteca de visualización. Esta herramienta permite visualizar cualquier Point Cloud o conjunto de estAs mediante la llamada por consola de:

pcl_viewer [point_cloud_1] [point_cloud_2]…

La herramienta de visualización permite al usuario elegir parámetros tan diversos como:

- Color de representación
- Gradiente de colores
- Zoom
- Rotación y traslación

Un ejemplo de visualización con las distintas características se proporciona a continuación:

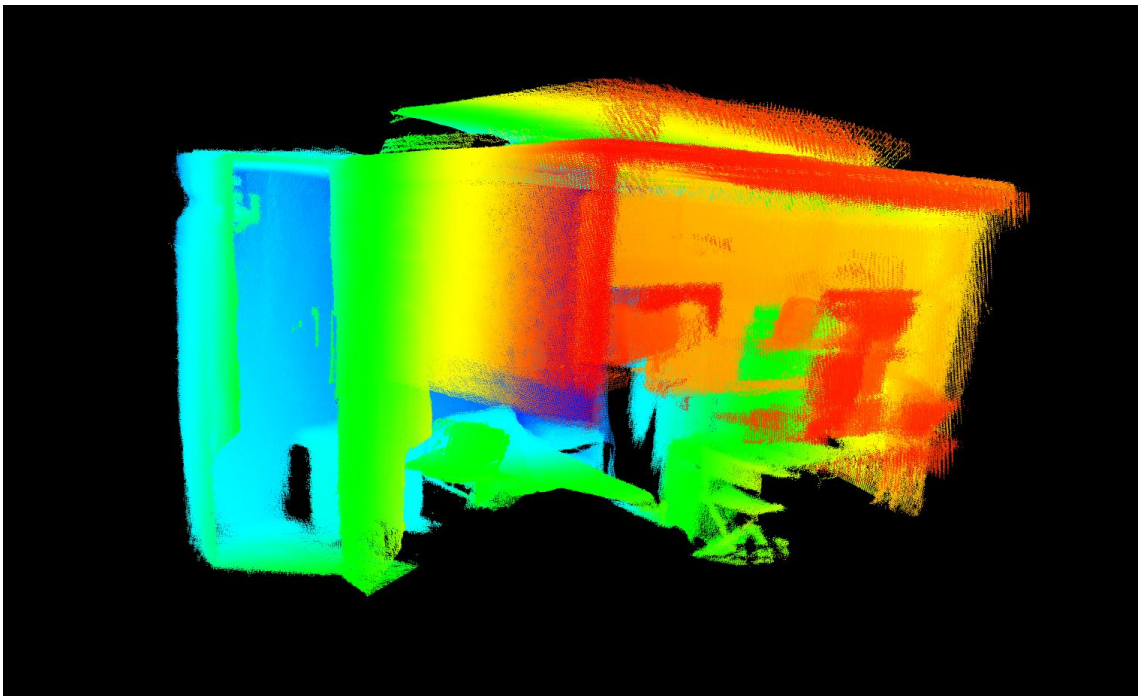


Ilustración 34 - Visualizador de Point Clouds PCL

Para comenzar, se procede con el filtrado de puntos. Esta pequeña sección del programa de tratamiento se encarga de simplificar el modelo. Dado un modelo original se aplica la técnica de “downsampling” indicada en capítulos anteriores.

**Extract_indices**

Esta parte del programa se encarga de obtener las superficies horizontales (con las que concluimos que se iba a realizar la representación en 2D del entorno para el robot). De un entorno dado:

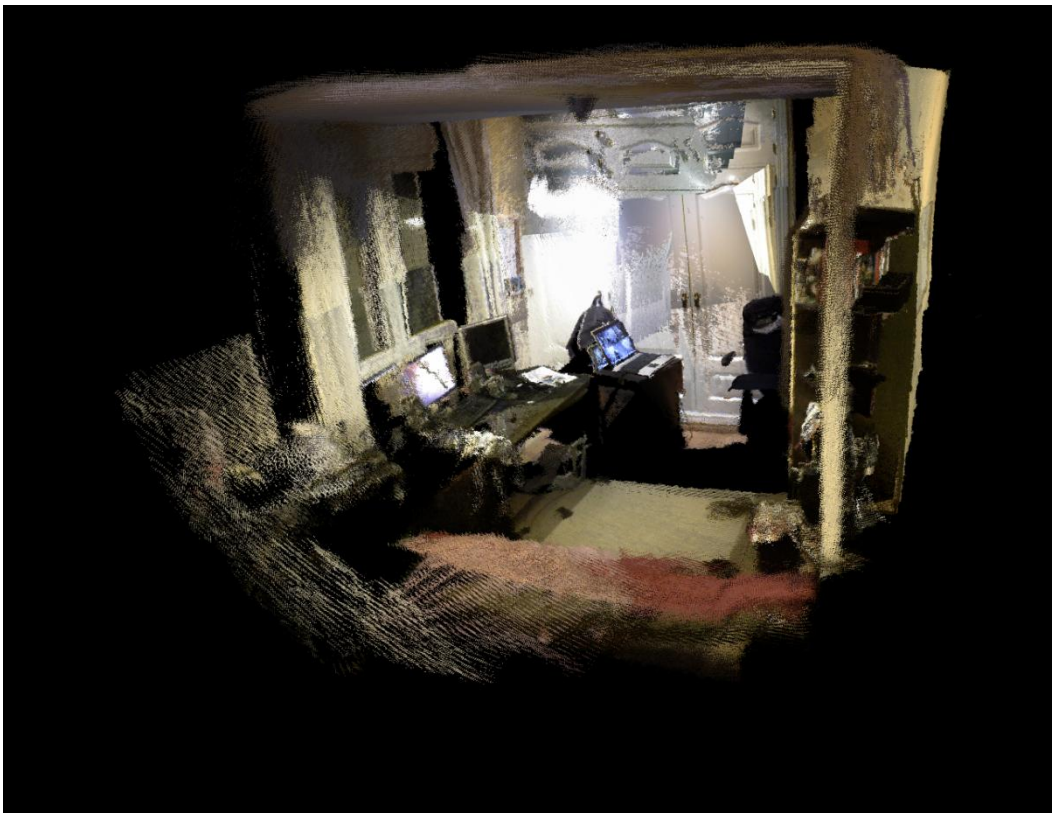


Ilustración 35 - Point Cloud con datos RGB genérica

Se obtienen de forma progresiva las distintas superficies representativas del modelo. Una ventaja es que el proceso crea las superficies mayores (que se corresponden con el techo, suelo, etc.) y que más nos interesan con prioridad respecto al resto. Con el objetivo de no obtener demasiadas Point Clouds, se ha fijado el coeficiente de puntos mínimos para la generación de estas superficies de manera que las más pequeñas y que no aportan valor añadido al estudio son ignoradas. A continuación se representan 3 superficies características del modelo proporcionado anteriormente:

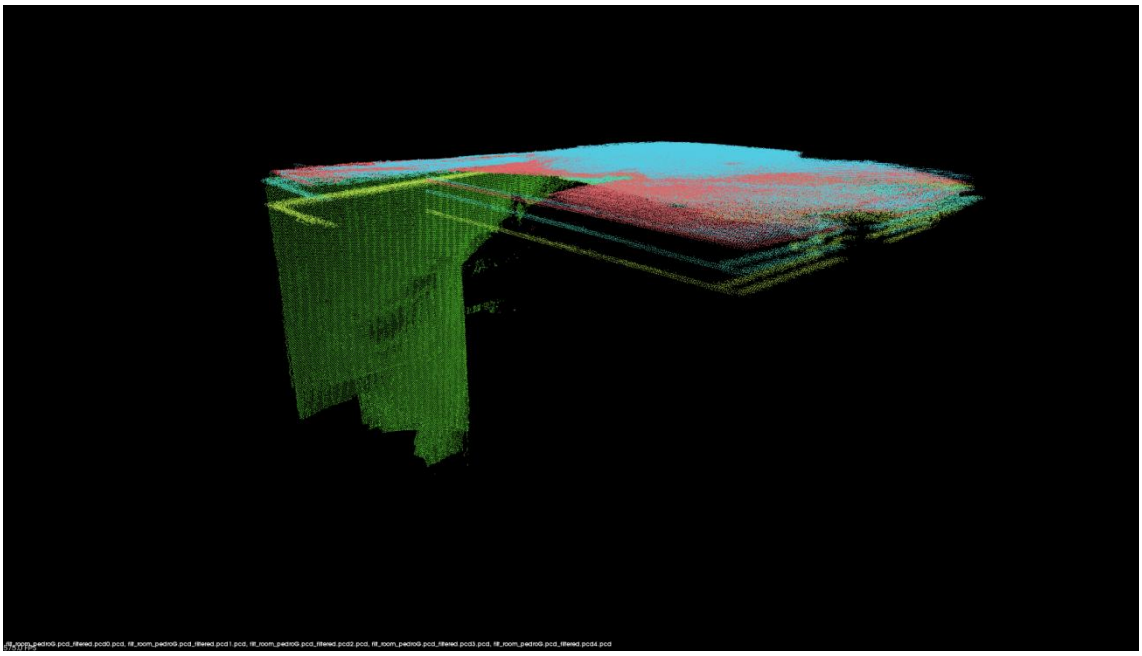


Ilustración 36 - Ejemplo de extracción de planos representativos de una Point Cloud (3 horizontales y 1 vertical)

### Concave_hull

Por último, y orientada a la obtención de los contornos de las superficies tratadas en el proceso anterior, aplicamos esta parte del código. Asumiendo que la mayor superficie obtenida en el paso anterior es el techo de la estancia, este último proceso se aplica solamente al archivo .pcd denominado como:

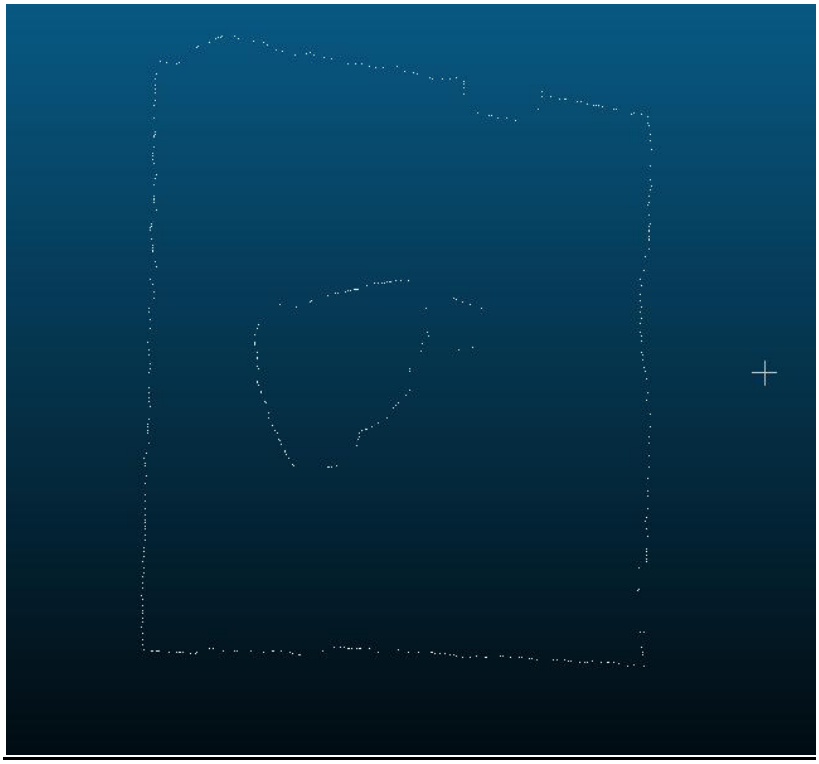


Ilustración 37 - Ejemplo de contorno extraído representado con CloudCompare para facilitar la visualización

### 4.1.1 Point Clouds obtenidas

Para desarrollar el proyecto, se ha elegido la planta superior de una vivienda. Dicho entorno de trabajo va a estar compuesto de los siguientes elementos, a cada uno de los cuales le corresponderá una o más Point Clouds dependiendo del tamaño o de la viabilidad de segmentar cada parte:

- Hall principal
- Dormitorio 1
- Dormitorio 2
- Dormitorio 3
- Dormitorio 4
- Baño 1
- Baño 2
- Baño 3

Como se ha detallado en la parte introductoria de la captura de datos, el primer mapa 3D generado por el método SLAM cuenta con una componente de color, que si bien a la hora de procesar las nube de puntos será obviada, es interesante observar para posibles experimentos posteriores. A continuación se adjuntan algunas de las estancias utilizadas en el trabajo.

### 4.1.2 Nomenclatura de las Point Clouds

Con el objetivo de clasificar y ordenar todas las Point Clouds asociadas al proyecto (se debe tener en cuenta no solo las nubes de puntos originales, sino también las obtenidas en los procesos intermedios del tratamiento), se ha decidido asignar la siguiente nomenclatura:

- Point Cloud original

  [Nombre de la habitación].pcd

- Point Cloud post-filtrado

  [Nombre de la habitación]_filtered.pcd

- Superficies

  [Nombre de la habitación]_filtered[Número del plano].pcd

- Contorno de la superficie

  [Nombre de la habitación]_filtered_0hull.pcd

Con este criterio se pretende agilizar la manipulación, visualización y control de las Point Clouds obtenidas a los largo de todo el proceso de la confección del mapa 3D global de cara al usuario.

### 4.1.3 Fuentes de error y limitaciones

Si bien este apartado se expandirá en las conclusiones del proyecto (y aplicando el concepto de error al conjunto global del proyecto), se deben puntualizar diversas fuentes de error en el proceso de tratamiento de las Point Clouds.

Uno de los conceptos mencionados con anterioridad es la necesidad de existencia de ciertos "landmarks" o puntos de referencia a la hora de aplicar SLAM. En estancias relativamente pequeñas (como han sido en su mayoría las empleadas en este proyecto), la ausencia de detalles o elementos de referencia en techo y paredes ha complicado en gran medida el proceso de tomar de datos, deteniéndose el software de captura al no encontrar una referencia intermedia entre 2 puntos diferentes del entorno a modelar.

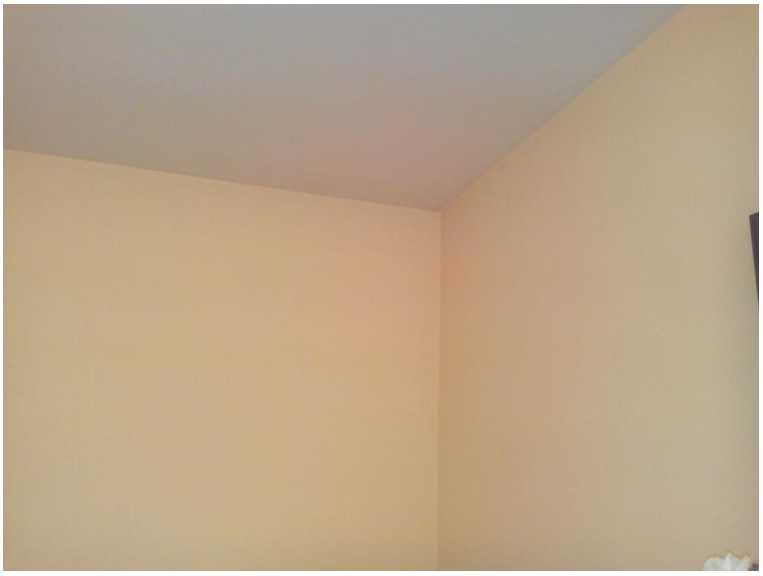


Ilustración 38 - Ausencia de elementos debido al mal posicionamiento de la cámada

Este factor ha condicionado la selección del lugar inicial de colocación de la cámara. La conclusión sería establecer una posición fija para la cámara en todos los entornos, y sería lógico asumir que esta posición viene definida por el centro geométrico de la estancia. Sin embargo, la riqueza de elementos candidatos a constituir un punto de referencia en ciertas estancias (estanterías, cuadros, etc) hace que con la intención de obtener el modelo más fiel y de la manera más cómoda posible, en ocasiones se posicione el sensor en un punto más cercano a las paredes con estos elementos, de manera que debido a la mayor distancia respecto a la región problemática, sea posible incluir un mayor número de elementos entre ésta y la posición del sensor, como se puede observar en la siguiente imagen:

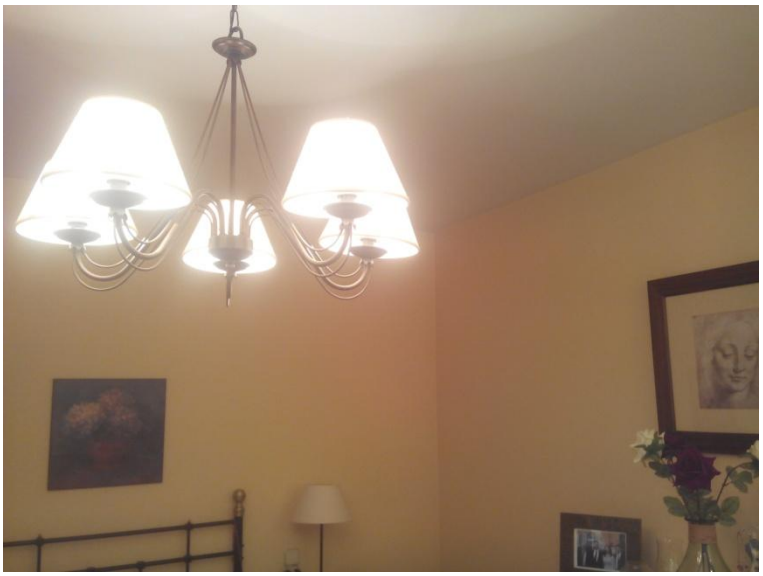


Ilustración 39 - – Al cambiar la posición, los elementos permiten la ejecución del algoritmo de matching respecto al siguiente fotograma.

Relacionado con esta última imagen, es necesario tener en cuenta que cualquier obstáculo (en este caso la lámpara) priva al sensor de la información que hay detrás de él. El usuario/robot debe de ser capaz de realizar los movimientos necesarios para evitar la aparición de estos puntos ciegos, que se traducirán en pérdida información como la que se puede observar en la siguiente imagen:

Ilustración 40 - Error producido por obstáculo

Se puede observar que la existencia de una lámpara provoca la pérdida de información en una región tan sensible para este proyecto como es la sección superior de una de las paredes laterales (hay que recordar que al realizar la captura de puntos ponemos especial interés en el cambio de superficie techo-pared).

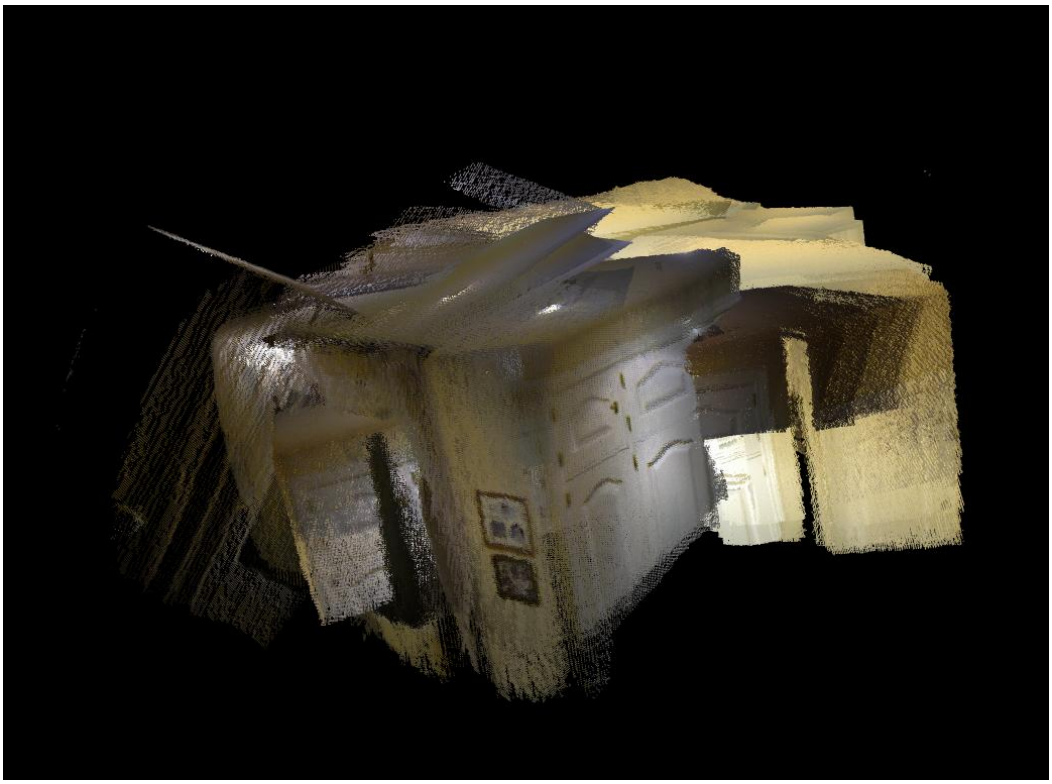


Ilustración 41 – Errores debido a movimientos bruscos del sensor

# 5. Resultados

El formato seguido para presentar los resultados va a ser el siguiente:

1. Point Cloud original con el número de puntos indicado
2. Point Cloud tras un primer filtro
3. Plano más representativo (techo) de la Point Cloud asociada
4. Contorno del plano más representativo

## 5.1 Metodología

A la hora de diseñar un proceso para la obtención de las nubes de puntos, se han considerado varias estrategias que luego se compararán con el objetivo de obtener la metodología que más se ajuste a los requerimientos de precisión y capacidad computacional en el proyecto.

Se debe considerar en todo momento el tiempo de procesamiento de los algoritmos necesarios, con lo que la velocidad de captura debe ser la adecuada.

Por otro lado, si bien el volver a evaluar ciertas regiones del entorno que ya han sido representadas en el mapa 3D previamente, es necesario considerar la posibilidad de solapamientos e incremento de imprecisiones en la toma de datos, especialmente si la cámara ha sufrido un cambio de posición respecto a la captura de datos previa asociada a esa zona.

Enlazando con las incertidumbres asociadas al movimiento de la cámara, es necesario ajustar la metodología dependiendo de la zona a tratar en cada caso. La existencia de zonas “ciegas” hace necesaria la manipulación de la cámara tanto en rotación como en traslación. Sin embargo, sería aconsejable ceñirse a la rotación de la cámara exclusivamente en la medida de lo posible.

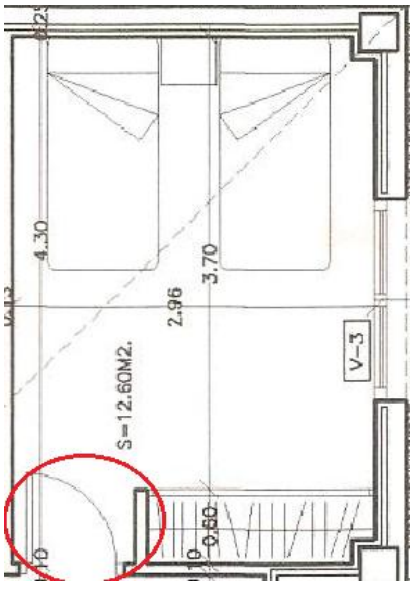



Ilustración 43 - Zonas problemáticas de las estancias

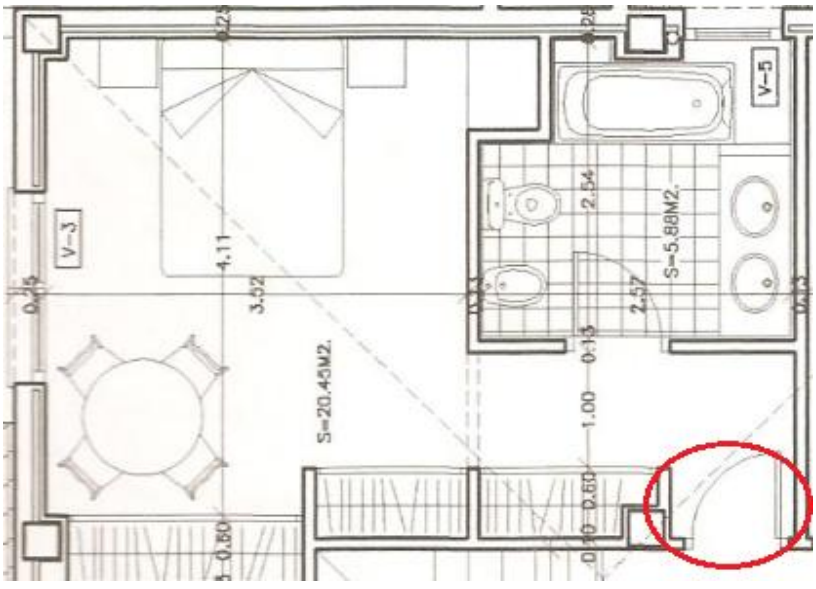
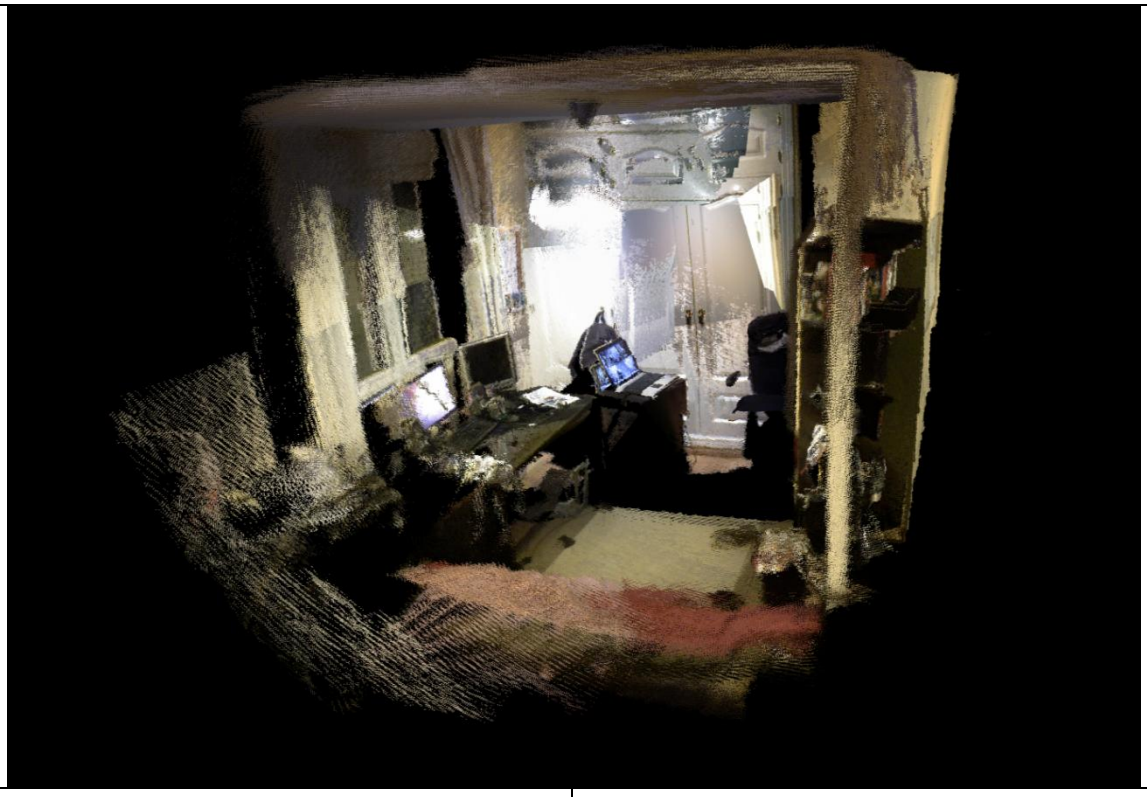
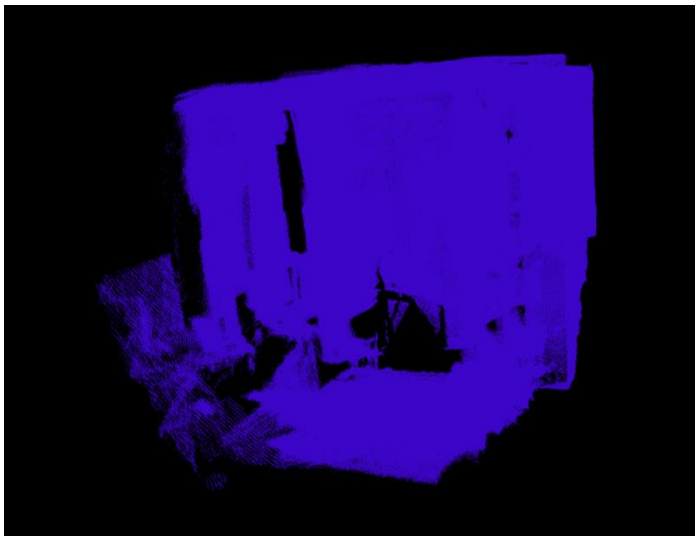
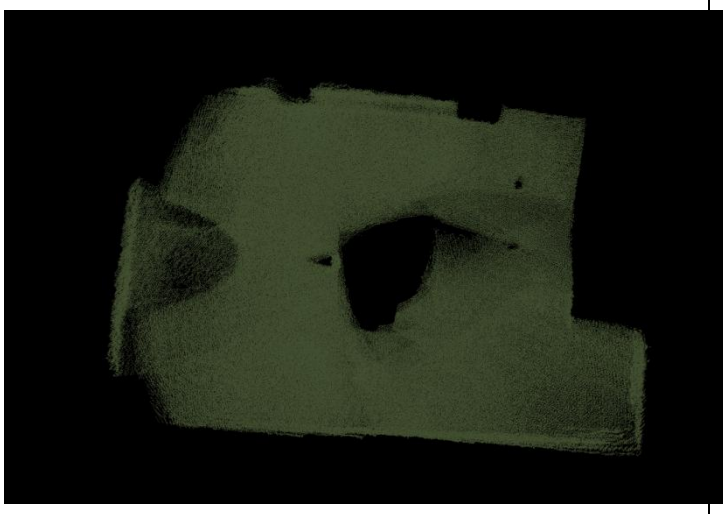



Ilustración 42 - Zonas problemáticas de las estancias II

**Dormitorio 1**

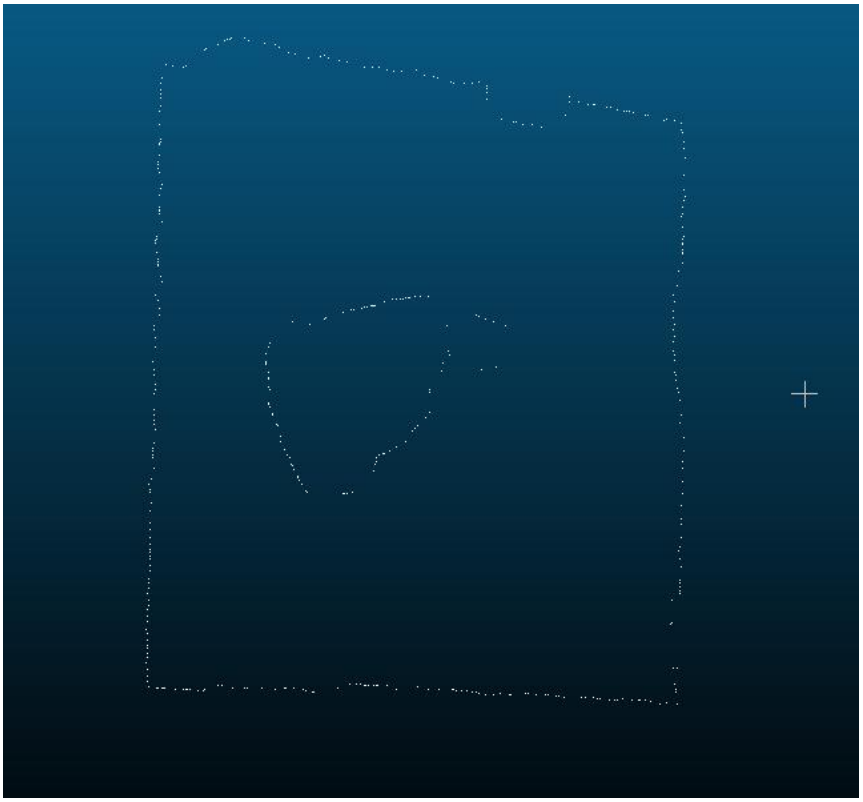


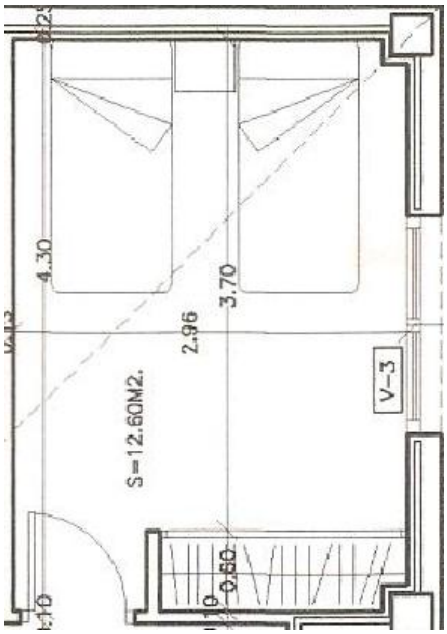




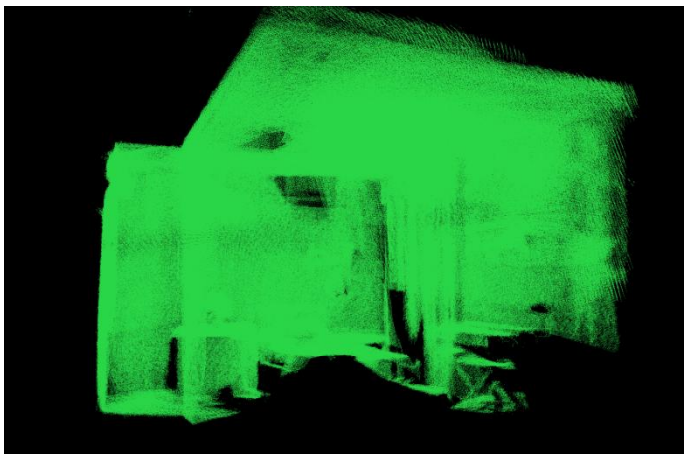
| Nombre | Número de puntos |
|---|---|
| Nube de puntos original | 16.823.635 |
| Nube de puntos filtrada | 2.290.838 |
| Plano | 146.905 |

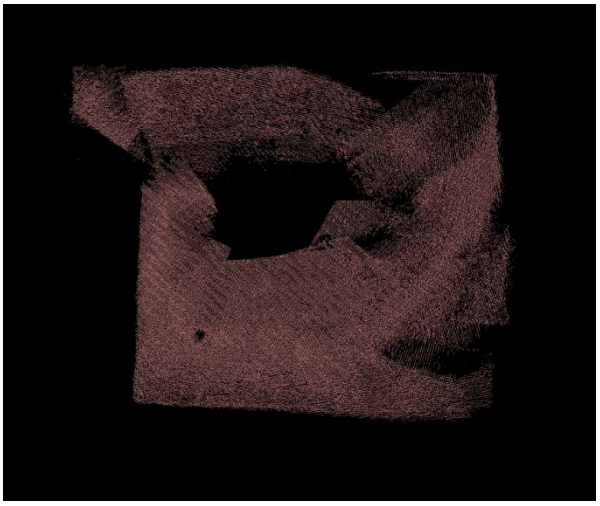
**Contorno de la habitación**



| Contorno | 508 |
|---|---|
| **Contorno teórico**<br>  | **Comentarios**<br><br>La sección de la puerta no se ha incluido en la generación del contorno.<br><br>Aunque aparece representada en el plano, el proceso de filtrado previo de dicho plano ha eliminado esa parte de la estancia. Esto es debido al error acumulado en esa zona debido al cambio de posicionamiento de la cámara requerido para cubrir toda la superficie de la puerta.<br><br>A su vez se observa pérdida de información en una parte del techo, si bien los bordes han permanecido intactos en su totalidad y por lo tanto se ha obtenido un perfil rectangular bastante preciso. |

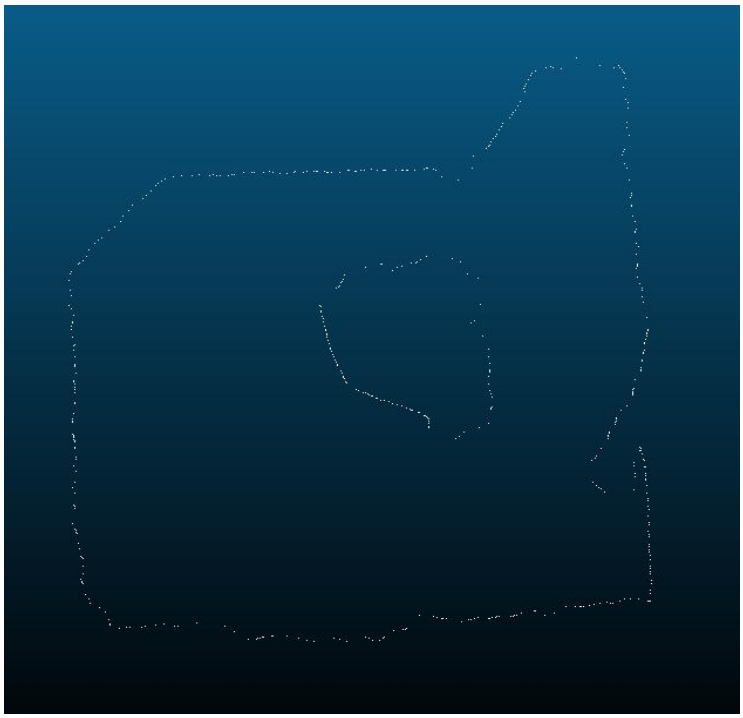
**Dormitorio 3**

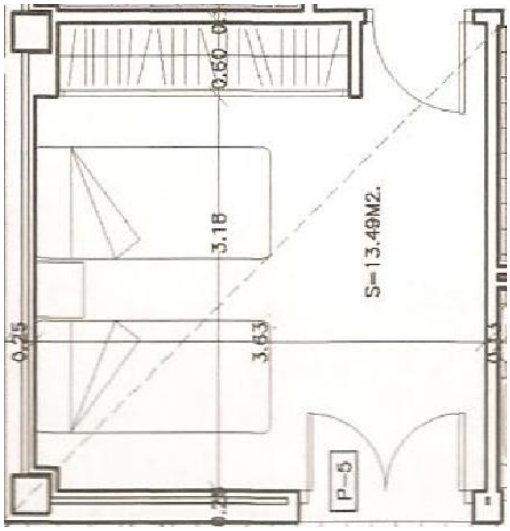




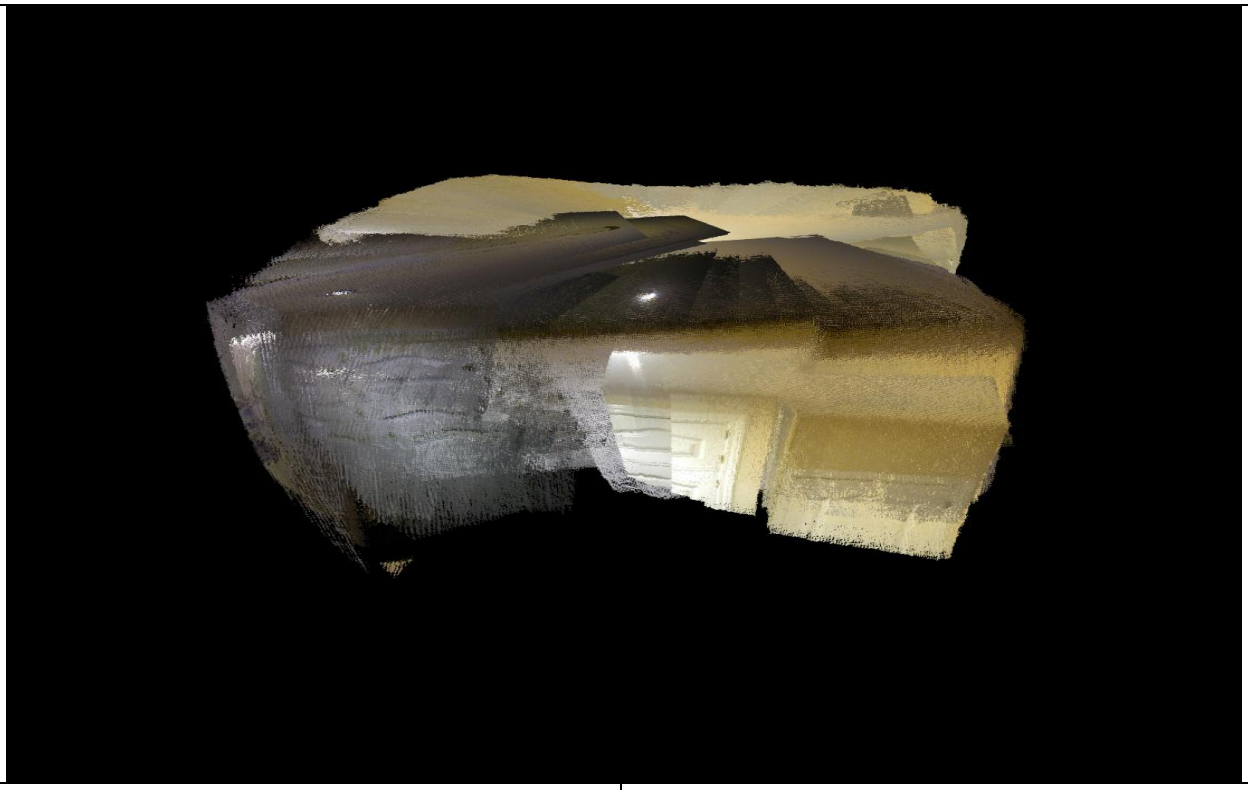
| Nombre | Número de puntos |
|---|---|
| Nube de puntos original | 15.904.479 |
| Nube de puntos filtrada | 3.262.983 |
| Plano | 138.521 |

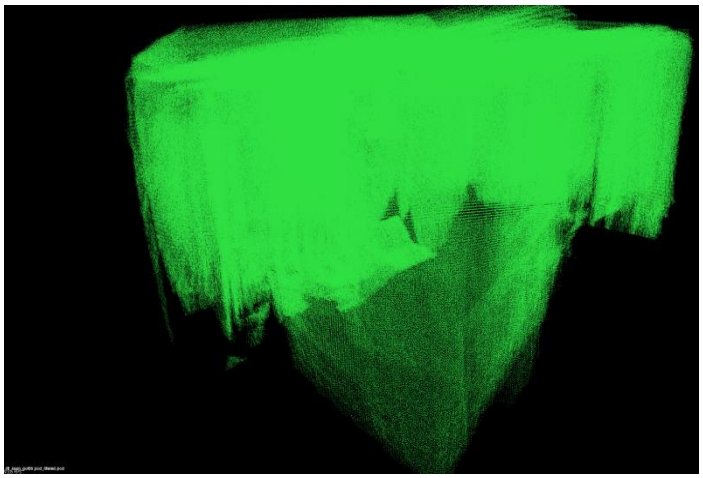
**Contorno de la habitación**

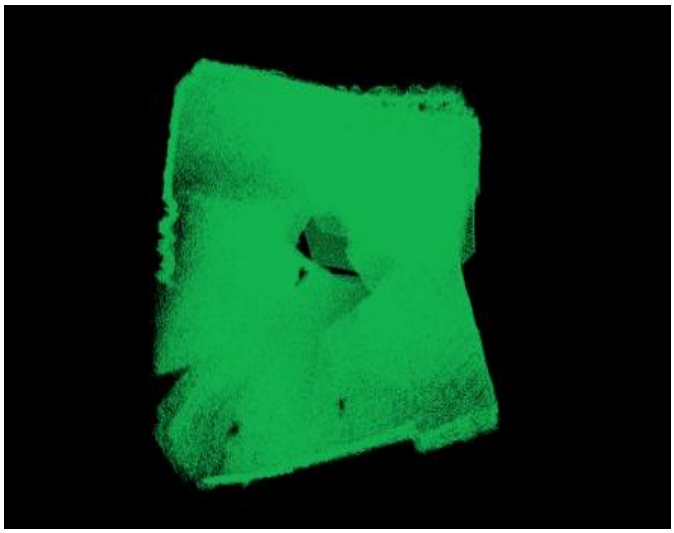


| Contorno | 498 |
|---|---|
| **Contorno teórico** | **Comentarios** |
|   | En esta ocasión, si se puede observar que la pequeña sección situada junto a la puerta si ha quedado parcialmente registrada en la obtención del contorno.<br><br>Se puede apreciar cierta pérdida de información en las esquinas, característica común a varias de las estancias. |

**Dormitorio 2**

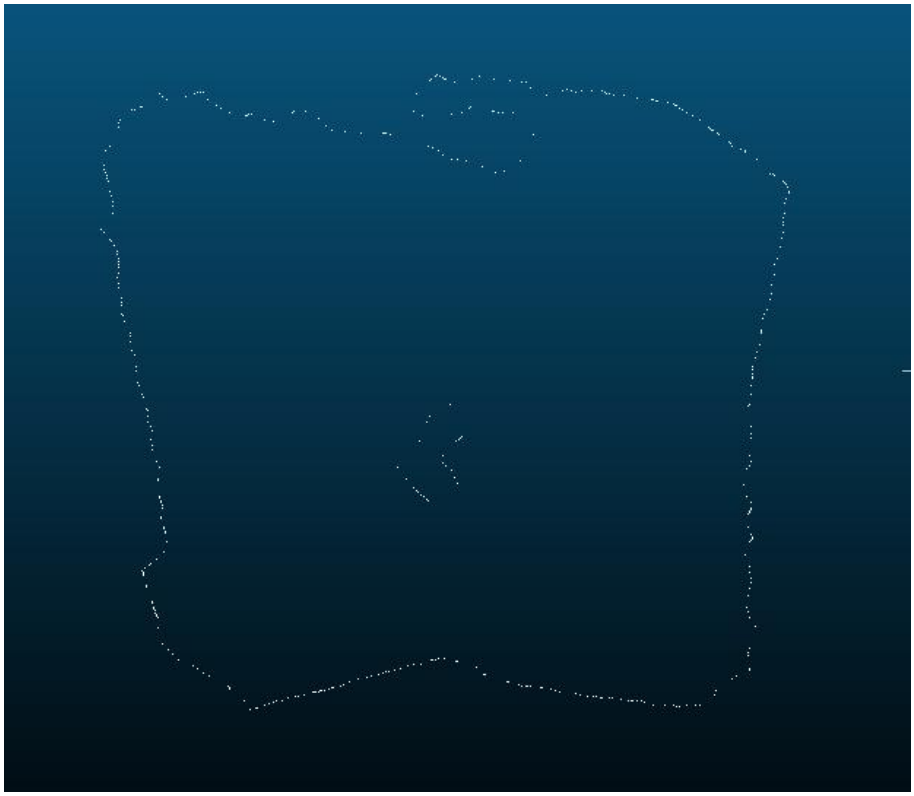


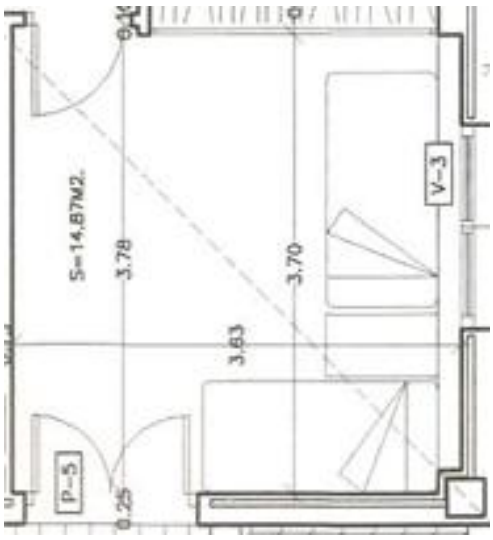




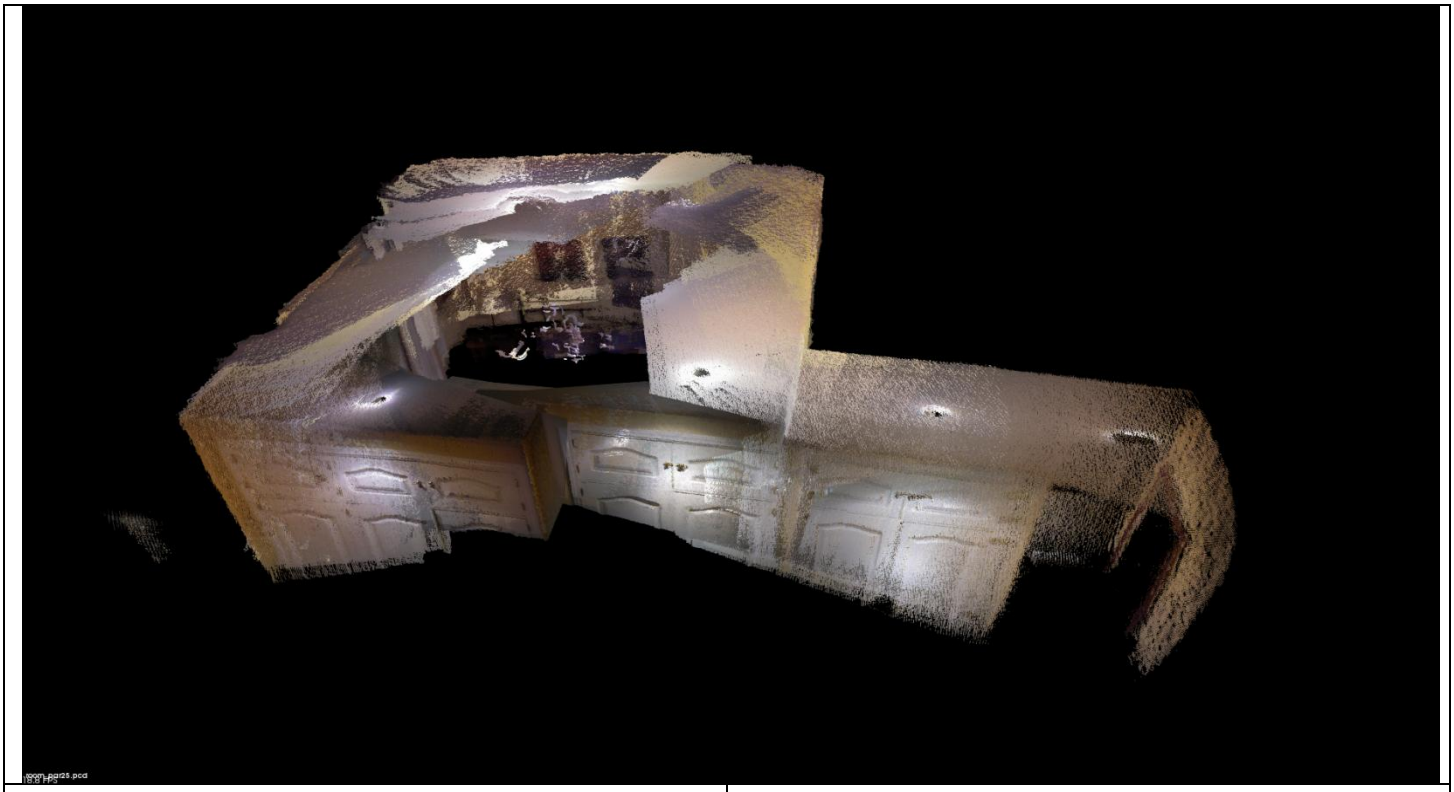
| Nombre | Número de puntos |
|---|---|
| Nube de puntos original | 13.280.262 |
| Nube de puntos filtrada | 3.934.360 |
| Plano | 677.839 |

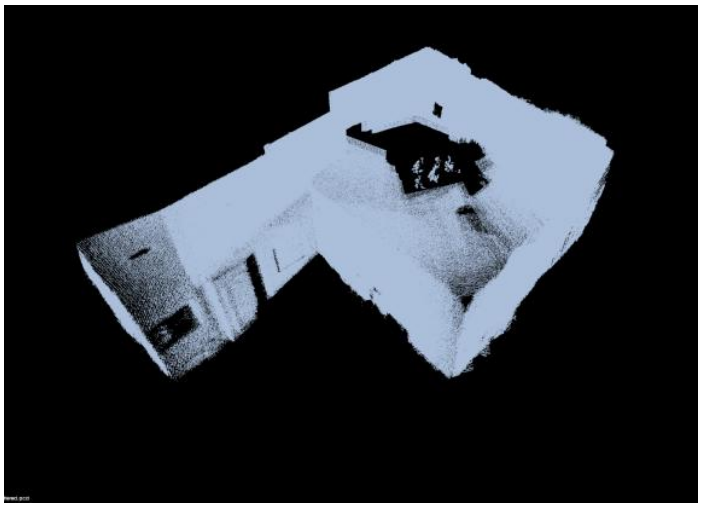
**Contorno de la habitación**

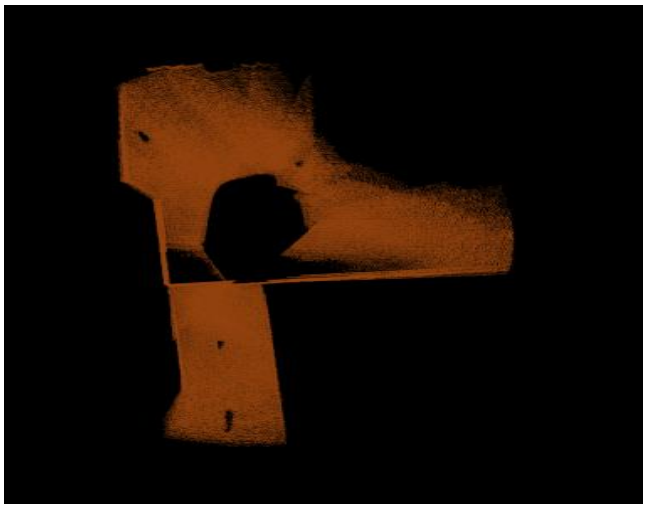


| Contorno | 413 |
|---|---|
| **Contorno teórico**<br>  | **Comentarios**<br><br>En este procesado se ha perdido la linealidad en las 4 esquinas de la habitación.<br>Sin embargo, vemos que la pérdida de información en el techo es menor que en otros ejemplos. |

**Dormitorio 4**

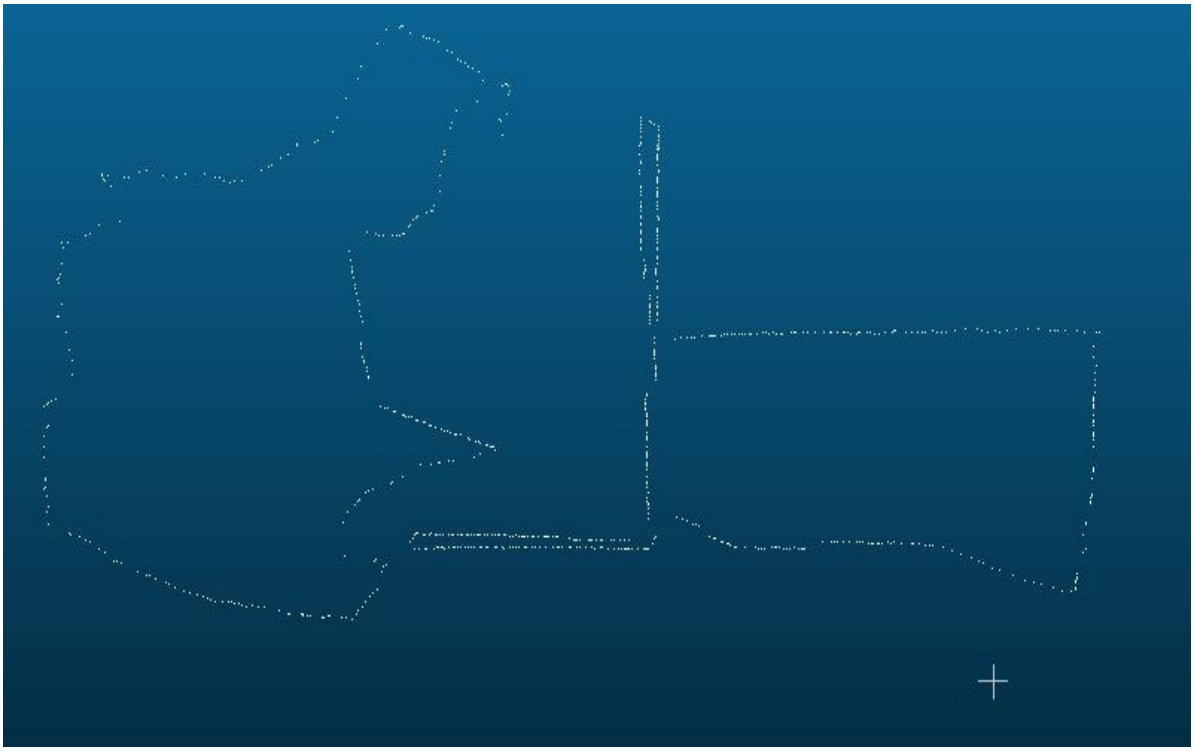


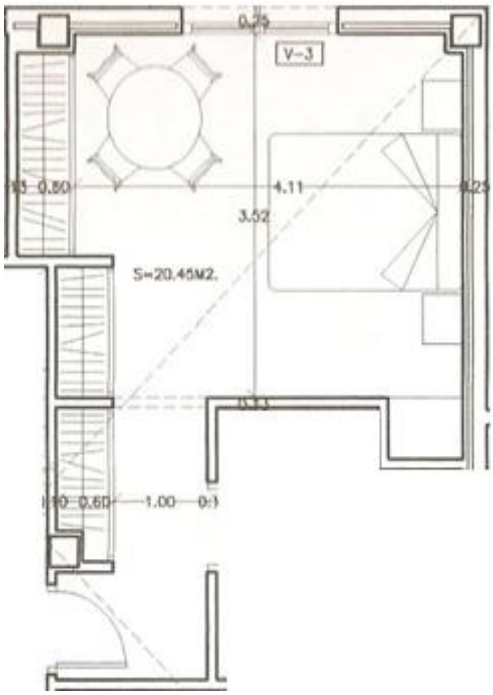




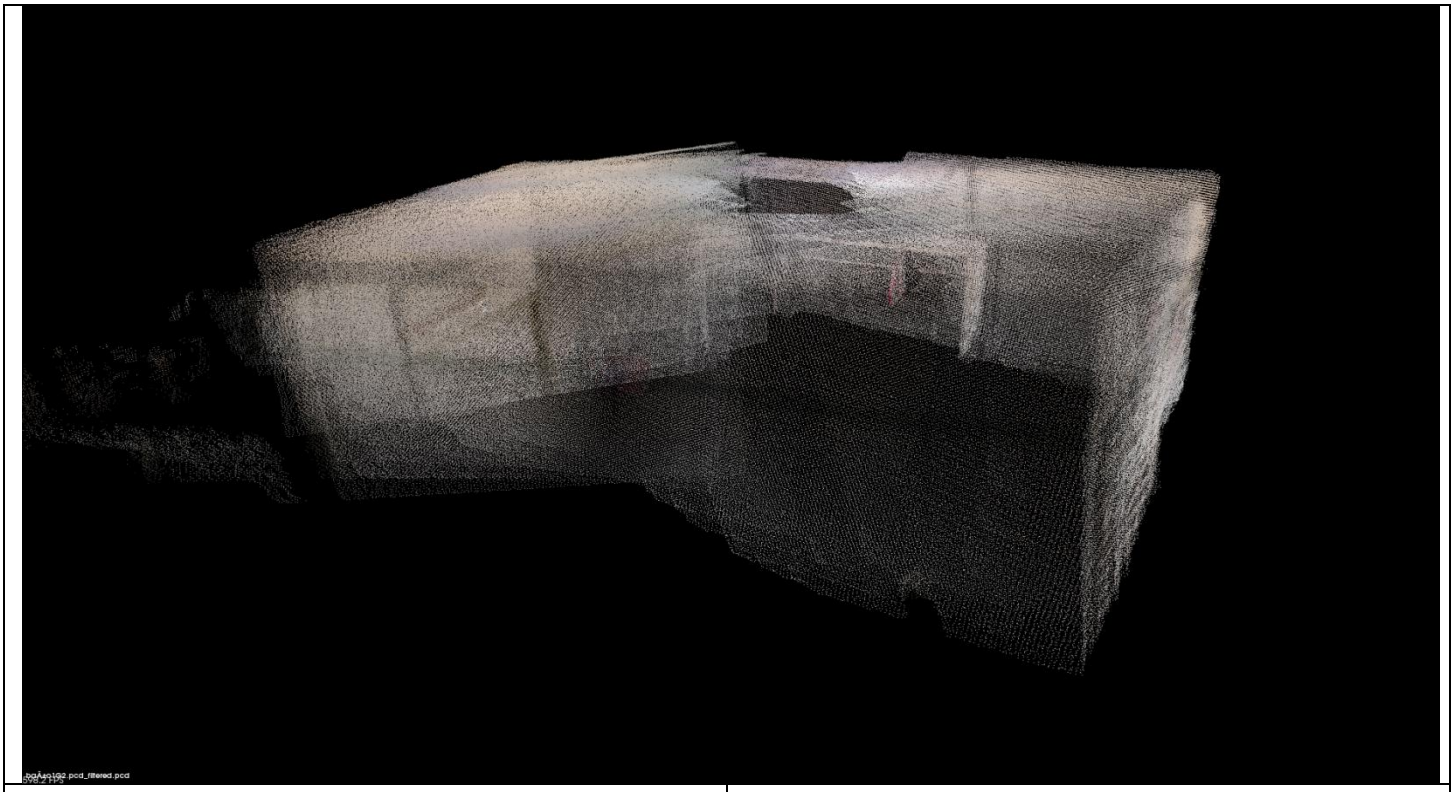
| Nombre | Número de puntos |
|---|---|
| Nube de puntos original | 9.137.979 |
| Nube de puntos filtrada | 2.214.950 |
| Plano | 304.235 |

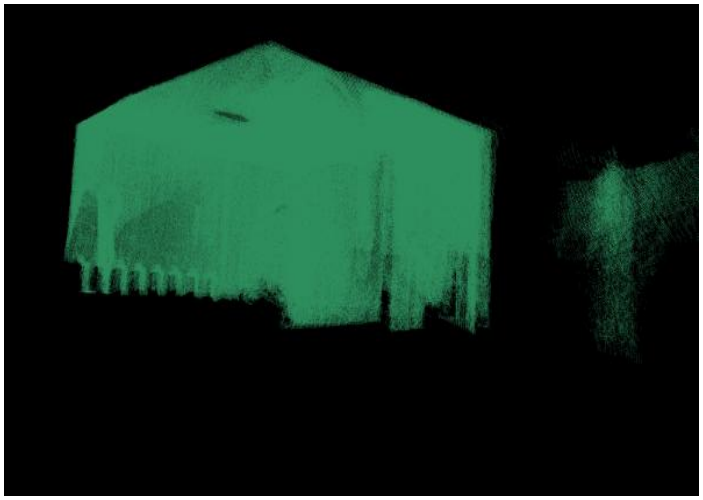
**Contorno de la habitación**

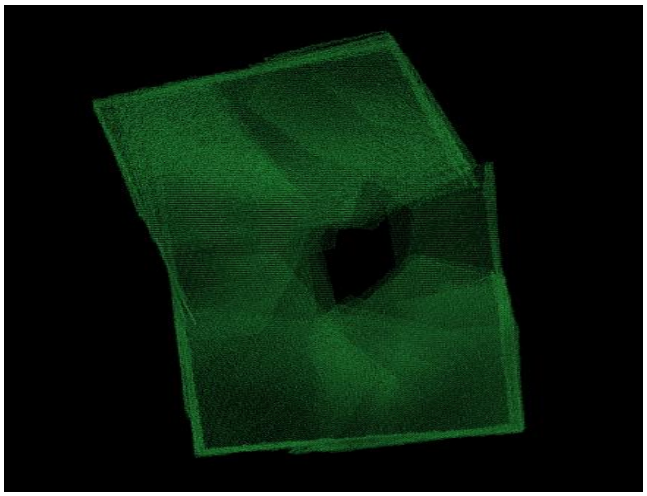


| Contorno | 799 |
|---|---|
| **Contorno teórico** | **Comentarios** |
|  | El Dormitorio 4 constituye una de las 2 secciones más críticas del proyecto. La existencia de un pequeño pasillo anexo a la estancia principal (y situada en un plano paralelo inferior al del resto de la habitación) genera el salto observado en el cambio de sección de la estancia.<br><br>A su vez, y pese a la agregación de landmarks improvisadas para realizar una mejora de los datos capturados durante más de 20 intentos, no se consiguió obtener un contorno representativo de la estancia libre de errores. |

**Baño 1**

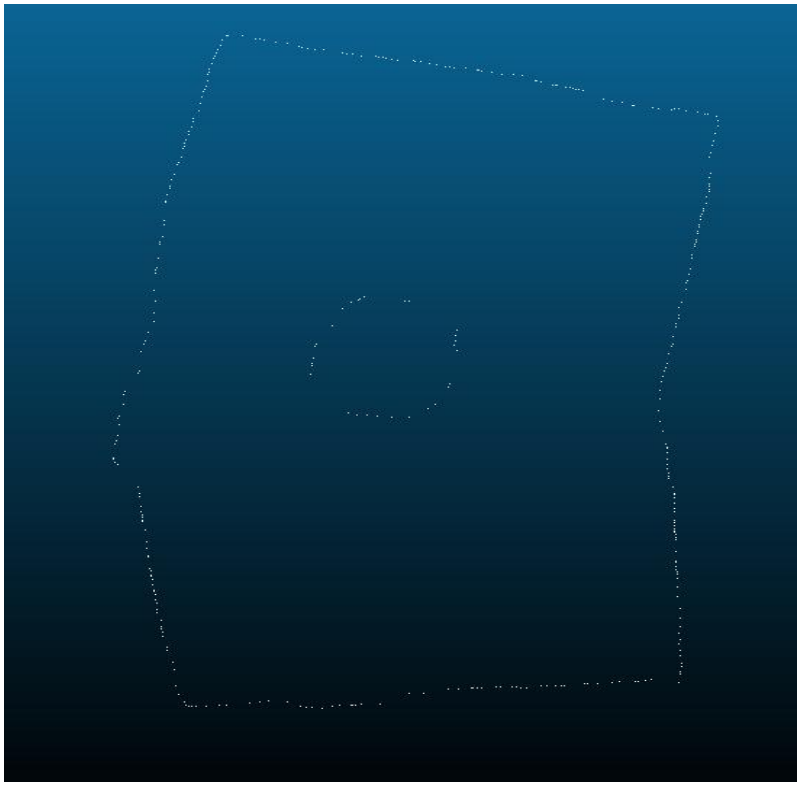


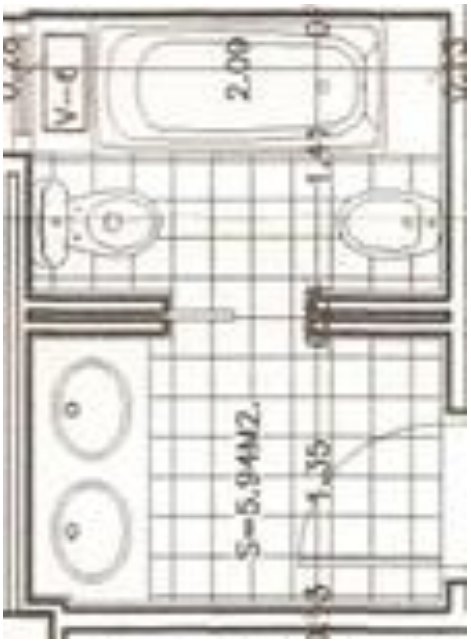




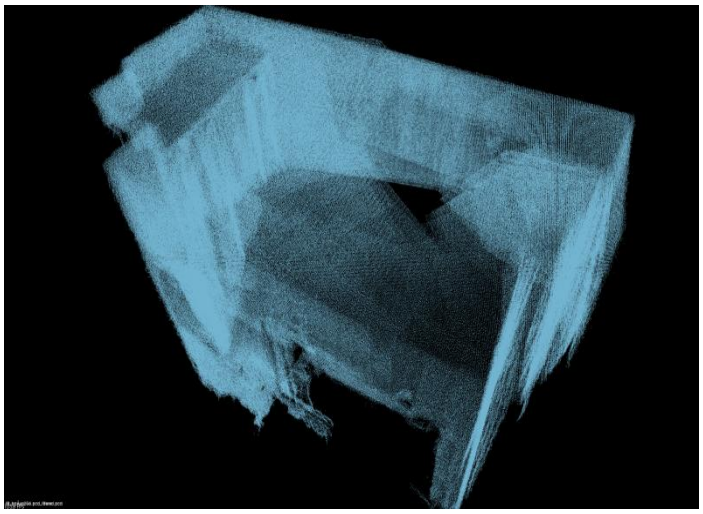
| Nombre | Número de puntos |
|---|---|
| Nube de puntos original | 7.162.320 |
| Nube de puntos filtrada | 489.965 |
| Plano | 188.434 |

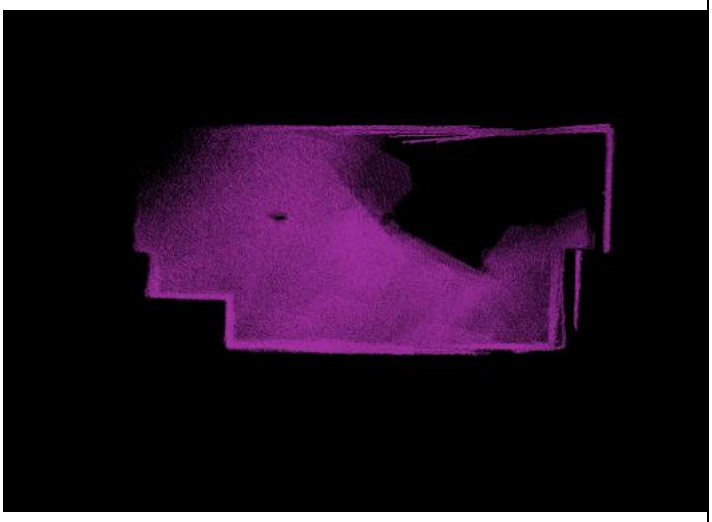
**Contorno de la habitación**



| Contorno | 360 |
|---|---|
| **Contorno teórico** | **Comentarios** |
|  | En este ejemplo se puede remarcar un error en el cambio de dirección de las paredes de la estancia durante el proceso de captura de datos con el sensor.<br><br>Considerando este aspecto, el contorno de la habitación resulta razonablemente regular. |

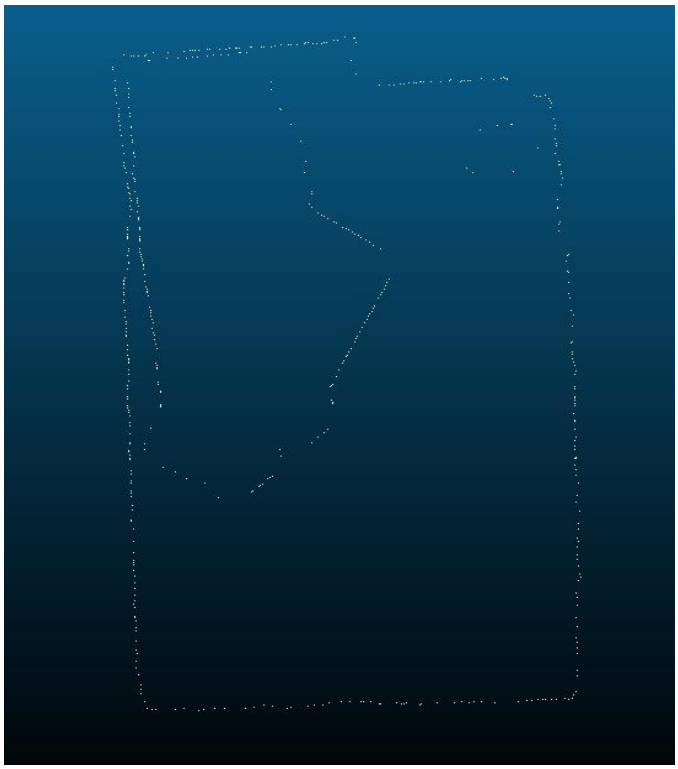
**Baño 2**

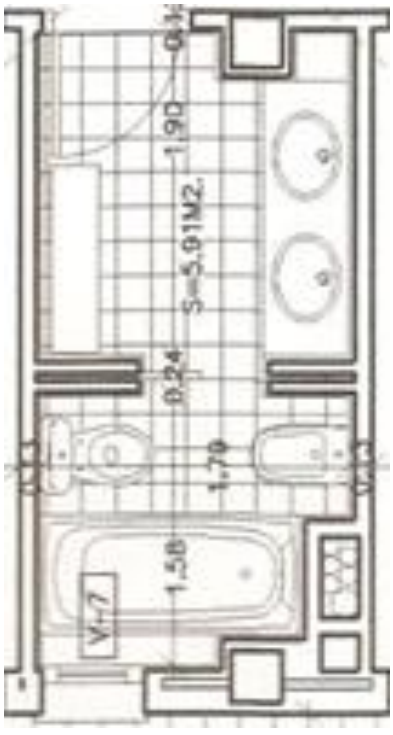




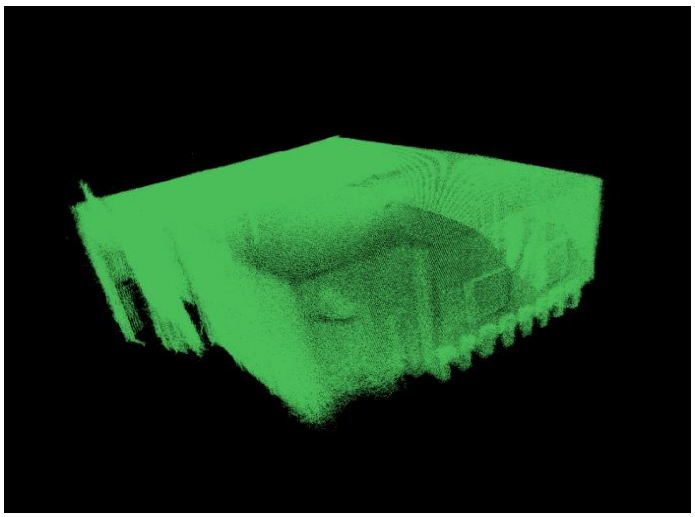
| Nombre | Número de puntos |
|---|---|
| Nube de puntos original | 7.804.914 |
| Nube de puntos filtrada | 1.298.428 |
| Plano | 248.025 |

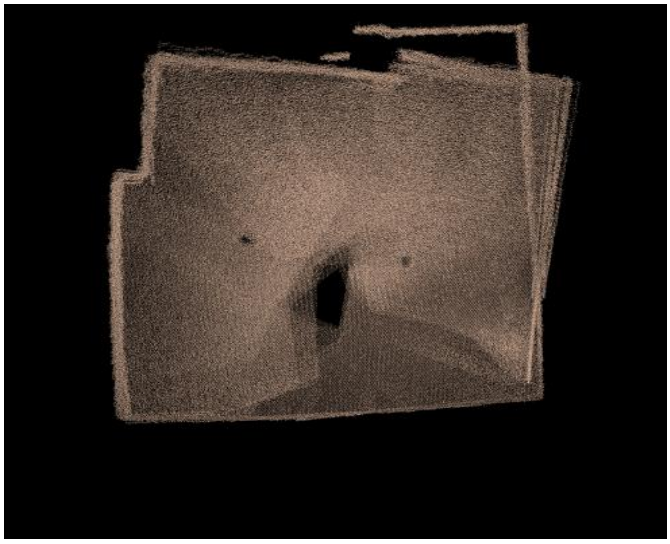
**Contorno de la habitación**



| Contorno | 539 |
|---|---|
| **Contorno teórico** | **Comentarios** |
|  | Para esta región en particular, observamos que pese a que se ha perdido una parte significativa de la información del techo, la zona correspondiente a la pared del recinto ha quedado intacta, permitiéndonos obtener un contorno mayoritariamente regular (otra vez con la existencia de irregularidades en la sección cercana a una de las esquinas de la habitación). |

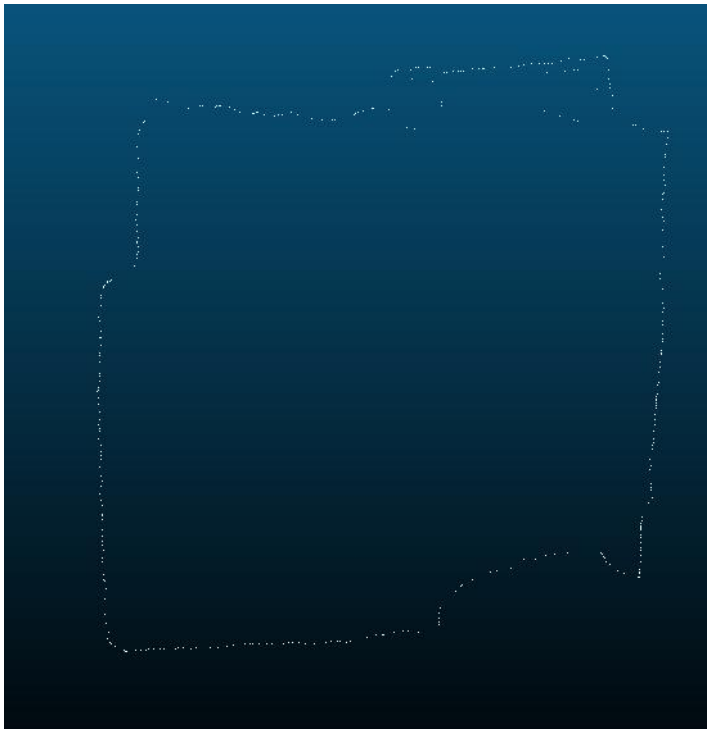
**Baño 3**

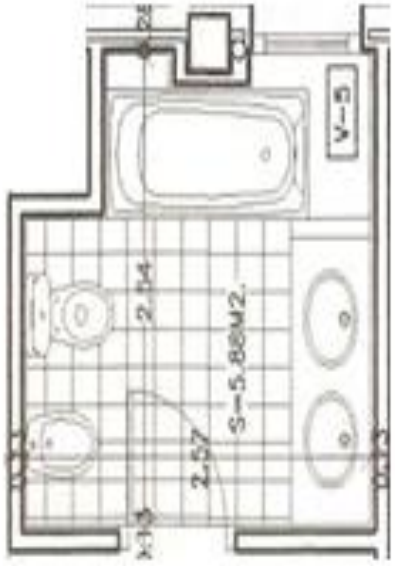




| Nombre | Número de puntos |
|---|---|
| Nube de puntos original | 6.051.029 |
| Nube de puntos filtrada | 1.063.506 |
| Plano | 326.100 |

**Contorno de la habitación**



| Contorno | 342 |
|---|---|
| **Contorno teórico**<br>  | **Comentarios**<br><br>La sección de la puerta no se ha incluido en la generación del contorno.<br><br>Aunque aparece representada en el plano, el proceso de filtrado previo de dicho plano ha eliminado esa parte de la estancia. Esto es debido al error acumulado en esa zona debido al cambio de posicionamiento de la cámara requerido para cubrir toda la superficie de la puerta.<br><br>A su vez se observa pérdida de información en una parte del techo, si bien los bordes han permanecido intactos en su totalidad y por lo tanto se ha obtenido un perfil rectangular bastante preciso. |

**Hall**

| Contorno | - |
|---|---|
| **Contorno teórico**<br>  | **Comentarios**<br>Como ya se ha comprobado de acuerdo al Dormitorio 4, la utilidad de los datos obtenidos en el proyecto decae proporcionalmente a la estancia capturada en cada momento.<br><br>Debido a la naturaleza del Hall (extensa superficie y gran ausencia de elementos de referencia en las mismas) los resultados obtenidos han sido menos representativos que los asociados al Dormitorio 4 y se omitido su inclusión en este apartado.<br><br>Sin embargo, la situación del Hall nos permite definir que constituye la región comprendida entre las demás estancias del contorno. |

## 5.2 Mapa topogeométrico

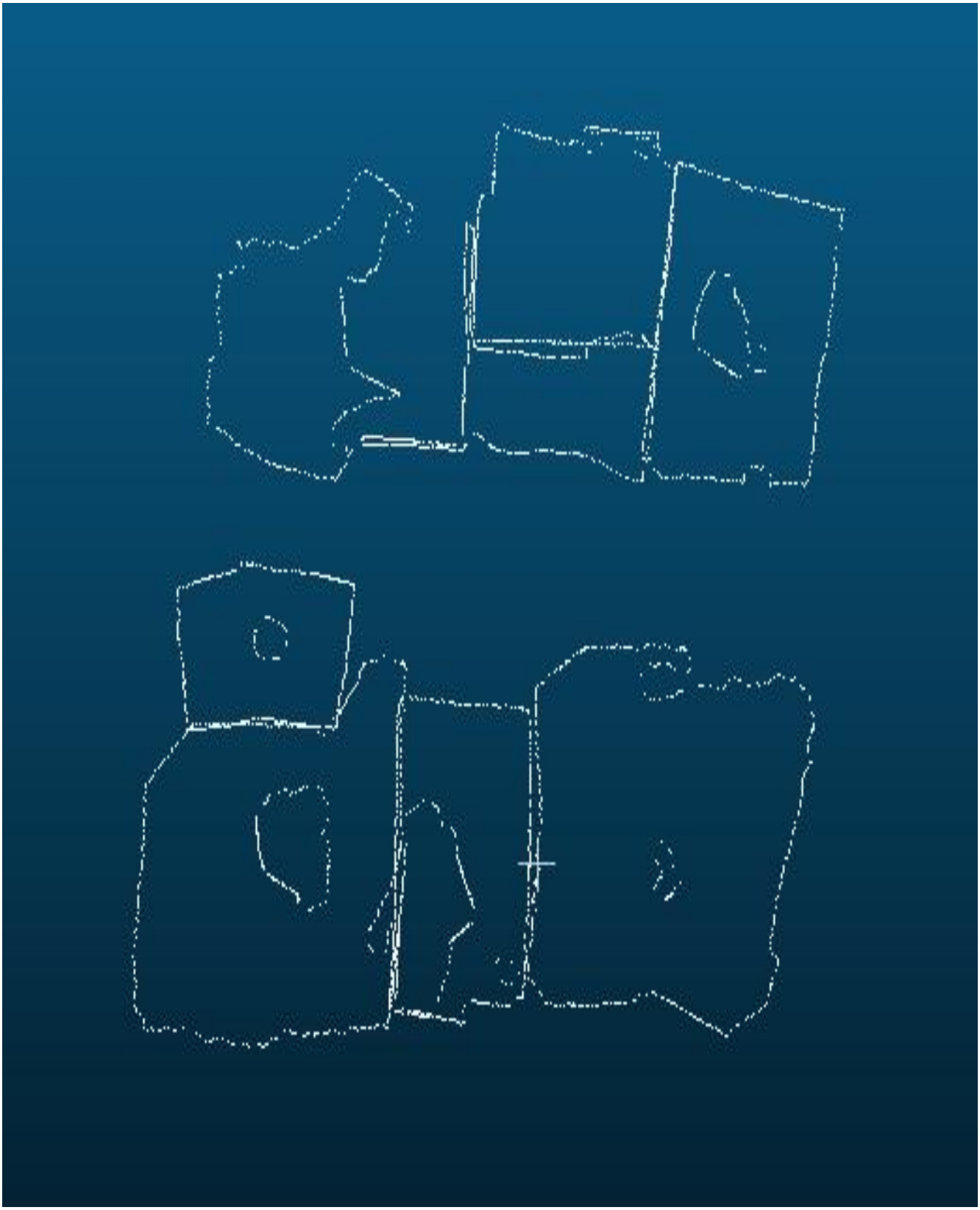


Ilustración 44 - Mapa topogeométrico obtenido

## 5.3 Mapa topogeométrico teórico

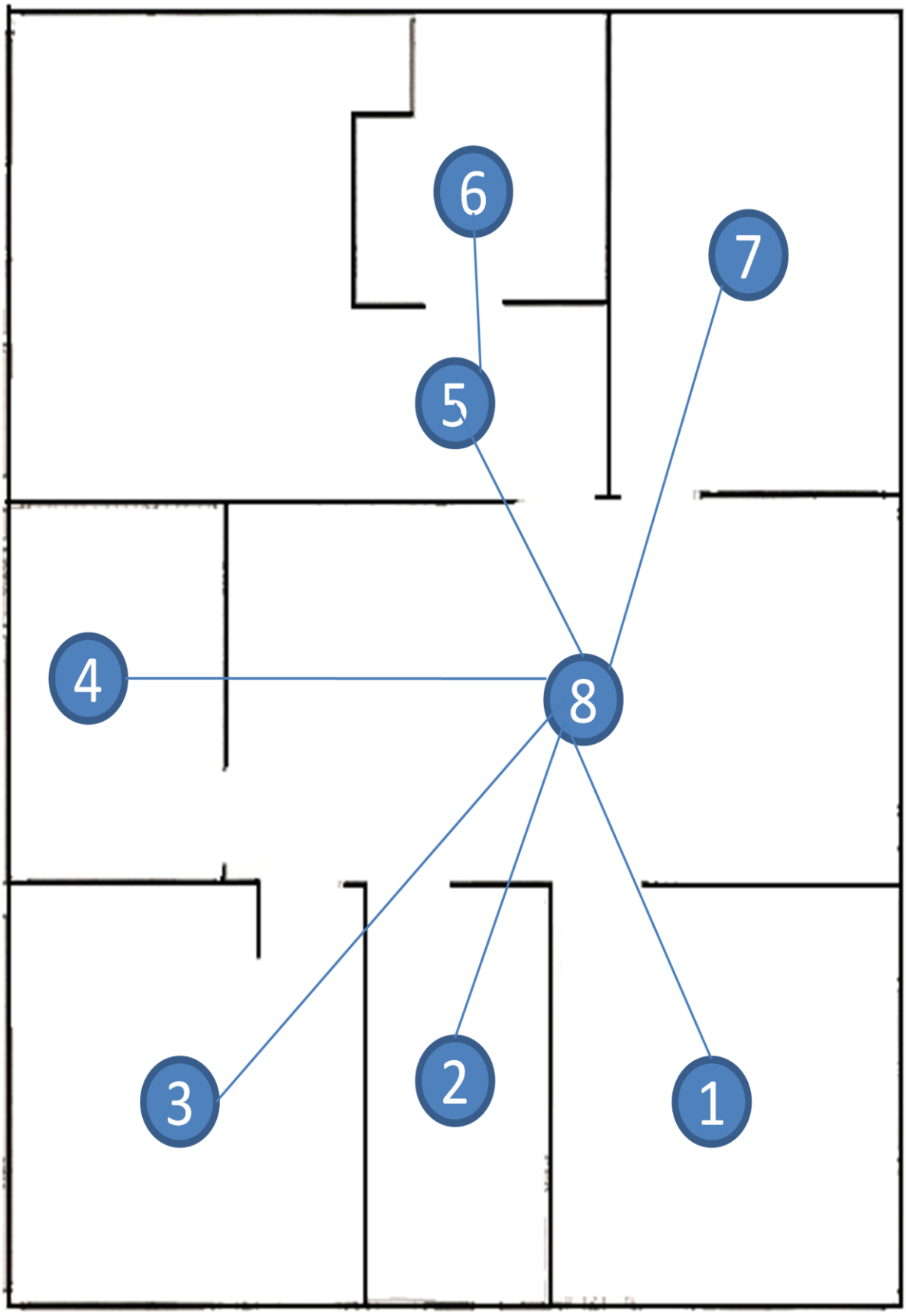



Ilustración 45 - Mapa topogeométrico real

Con la numeración establecida, la matriz asociada al grafo topogeométrico es la siguiente:

$$\begin{pmatrix} 0 & 2 & 2 & 2 & 2 & 3 & 2 & 1 \\ 2 & 0 & 2 & 2 & 2 & 3 & 2 & 1 \\ 2 & 2 & 0 & 2 & 2 & 3 & 2 & 1 \\ 2 & 2 & 2 & 0 & 2 & 3 & 2 & 1 \\ 2 & 2 & 2 & 2 & 0 & 1 & 2 & 1 \\ 3 & 3 & 3 & 3 & 1 & 0 & 3 & 2 \\ 2 & 2 & 2 & 2 & 2 & 3 & 0 & 1 \\ 1 & 1 & 2 & 2 & 1 & 2 & 2 & 0 \end{pmatrix}$$

## 5.4 Otros resultados

Como objetivo complementario al mapa topogeométrico, se había establecido la capacidad de asociar modelos 3D de cada habitación a su parte correspondiente del mapa con el fin de comprobar la viabilidad de obtener información detallada de la misma. Se muestran ejemplos obtenidos por el procedimiento SLAM:

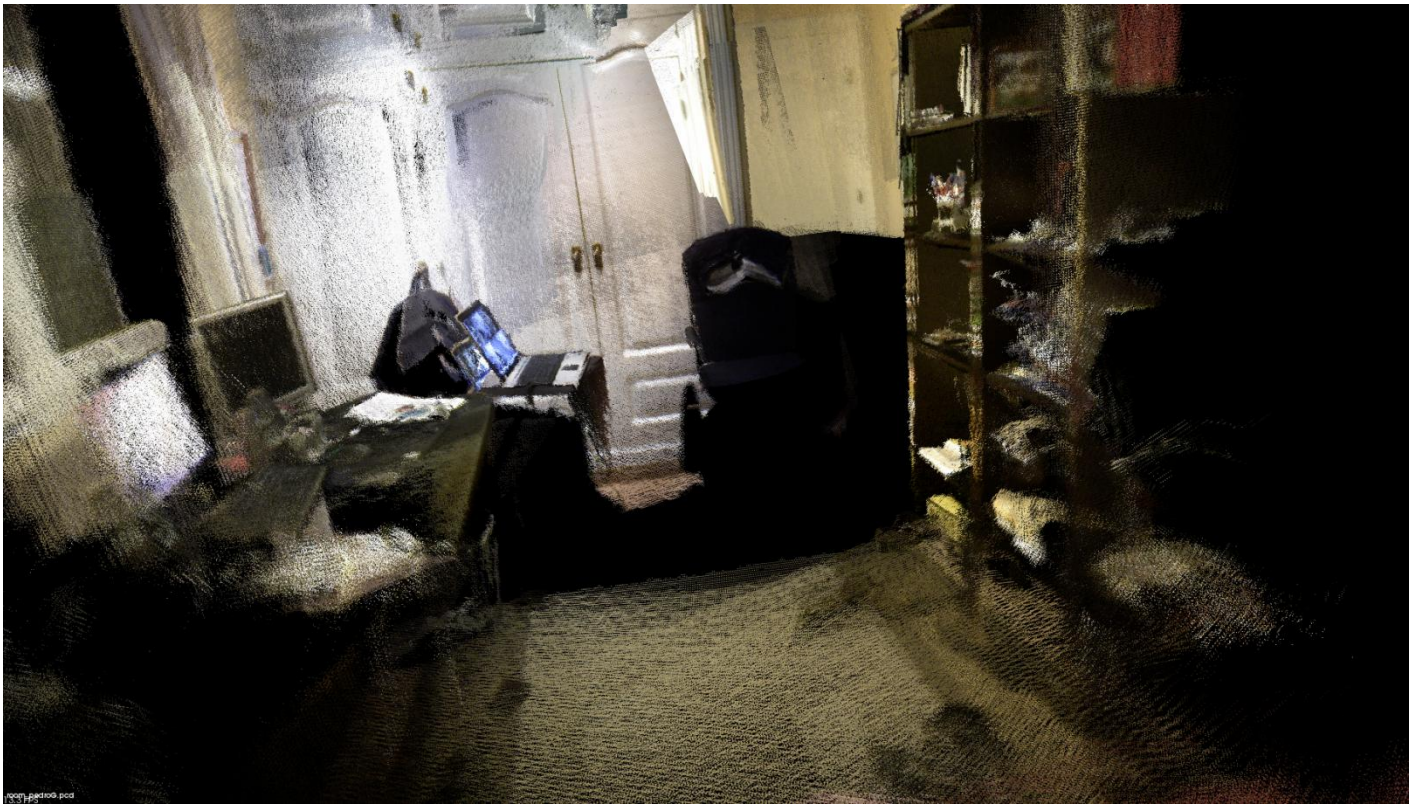


Ilustración 46 - Ejemplo 3D-RGB Dormitorio 1

Ilustración 48 - Ejemplo 3D-RGB Baño 3

Ilustración 47 - Ejemplo 3D-RGB Dormitorio 3

## 5.5 Valoración de los resultados

Como primera valoración, hay que interpretar los resultados del mapa topogeométrico obtenido teniendo en cuenta las limitaciones del procedimiento utilizado.

Se puede observar una equivalencia relativa entre el mapa teórico y el real. La posibilidad de numerosos errores en el mapa obtenido estaba muy presente a la hora de conducir los experimentos en cada estancia.

Uno de los factores que más destacan en el mapa final obtenido es la no linealidad de los contornos de las habitaciones capturadas. Esto viene determinado por la funcionalidad del algoritmo de extracción de planos y del contorno empleado, que no contempla la capacidad de dotar a la figura resultante una forma geométrica definida a partir de la forma general obtenida (generalmente de tipo rectangular).

Otro factor que vemos que ha influido significativamente es el tamaño de la estancia considerada. Por un lado tenemos una limitación ligada al rango efectivo de uso de la cámara ASUS, y por otro lado tenemos la incertidumbre y los errores acarreados en el procesamiento del software. La combinación de estos provoca que en los casos de las estancias Hall y Dormitorio 4, la forma general quede mucho menos definida, y las irregularidades obtenidas afecten a la claridad e interpretación del mapa final.

A continuación, hay que tener también en cuenta la escala a la que han quedado reducidos los compartimentos con respecto a su tamaño real dentro de la planta, que en ciertos

Dejando a un lado los errores que puedan haber sido provocados por las limitaciones del equipo, también hay que dedicar un análisis independiente a los errores humanos cometidos durante el procedimiento de obtención de resultados.

Se han efectuado distintas metodologías para intentar estandarizar la forma de capturar las Point Clouds.

Inicialmente se intentó mantener fijo el dispositivo y rotar este desde una posición constante en el espacio (en coordenadas xyz). Este método proporcionaba una rapidez mayor, pero conllevaba la pérdida de información y demostró no ser viable para la aplicación considerada en este proyecto. El problema radicaba en la no homogeneidad en la distribución de elementos de referencia en la estancia. Si bien es verdad que para ciertas regiones este método era apropiado, en algunos puntos con una densidad de “landmarks” menor, no existía la posibilidad de buscar otro sistema de referencia propio con una mejor perspectiva de la habitación.

Además, se imposibilitaba la capacidad de obtener información de regiones del espacio situadas detrás de algún obstáculo o elemento no deseado.

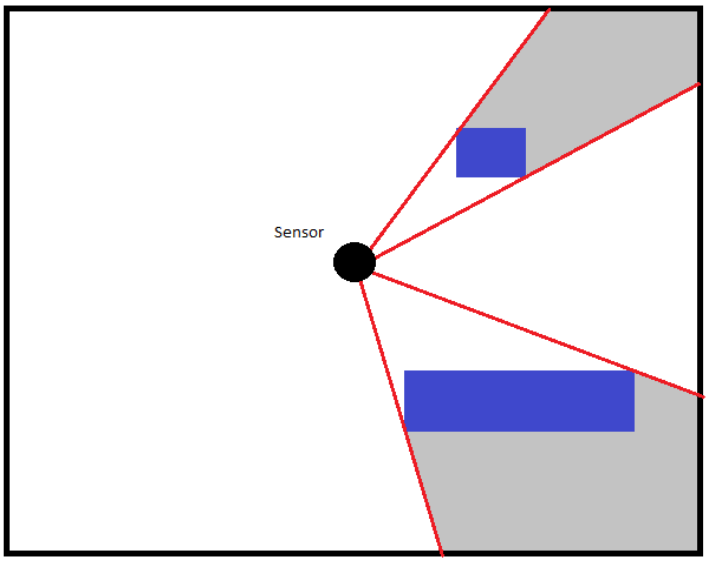



Ilustración 49 - Efecto de obstáculos para cámara fija

A continuación, se estudió la posibilidad de realizar un movimiento circular por la estancia, de manera que se enfocase la zona diametralmente opuesta a la ocupada por el sensor en la circunferencia constituida por su recorrido.

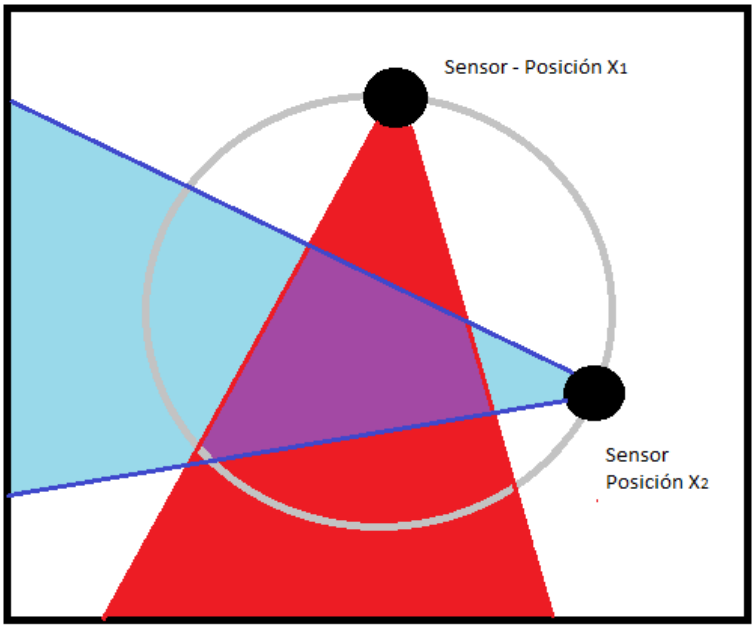



Ilustración 50 - Área cubierta por el sensor en trayectoria circular

Esta práctica demostró proporcionar mejores resultados, pero a su vez dependía también de la distribución de los posibles obstáculos repartidos por la estancia. Si bien no se ha llegado a aplicar el concepto de una trayectoria circular perfecta, leves variantes de esta metodología se han utilizado para la captura de datos asociados a varios de los mapas locales

Sin embargo, esta solución no ayuda a resolver el problema de pérdida de elementos de referencia necesarios para asegurar el correcto funcionamiento del algoritmo de asociación de fotogramas (explicado en capítulos anteriores).

Este último inconveniente deja 2 opciones:

a) Trayectoria personalizada dependiendo de la estancia

En relación a este trabajo, esta medida es ciertamente sencilla de aplicar. El número de estancias es pequeño y permiten al usuario realizar un escaneo teniendo en cuenta tanto la distribución de elementos de referencia como la distribución de obstáculos que limiten la disponibilidad de una trayectoria definida.

b) Agregación de "landmarks"

De la misma manera, esta solución viene sujeta a la existencia de un número limitado de estancias. La colocación de elementos de referencia estratégicos se ha empleado en la captura de ciertas de las estancias. Una de las principales razones por las que la limitación en la captura de puntos ha venido motivada por el problema de falta de elementos de referencia es la decisión de utilizar el cambio pared-techo como punto de partida para el proceso de extracción de superficies y contornos.

Generalmente, esta región está menos densamente poblada de elementos que podrían constituir puntos de referencia en SLAM. Por otro lado, pese a que el cambio de plano pared-suelo representa una región más atractiva a la hora de considerar estos "landmarks", la existencia de estos también es la causa de una pérdida de información asociada a la estructura propia de la habitación que se encuentra tras estos.

Con todo, es necesario considerar que la forma en la que se ha realizado la captura de los datos ha influido mucho en el resultado final de los mismos.

Extrapolando este problema a una situación con robots reales, sería necesaria la implementación de algoritmos que permitan al robot buscar alternativas en caso de pérdida de elementos de referencia, si bien el carácter generalmente terrestre (y por tanto careciente de la perspectiva de la altura que se ha tenido en este proyecto) genera que la forma más factible de realizar un procedimiento equivalente sea utilizando e cambio pared-suelo, con las ventajas e inconvenientes anteriormente descritos.

Por último, cabe destacar la existencia de una zona "ciega" alrededor de la zona central de las estancias.

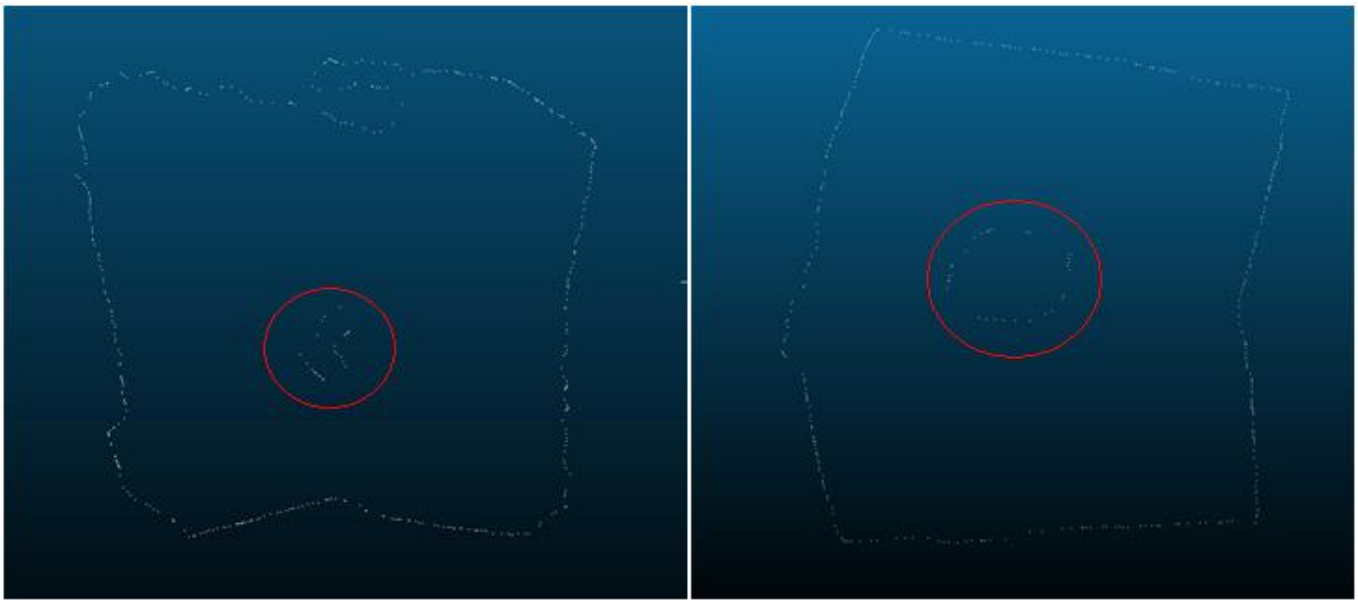


Ilustración 52 - Falta de información en el techo del recinto

Este fenómeno es debido a la imposibilidad de obtener una perspectiva que permita relacionar esta región del techo con elementos de referencia de la estancia. Aun en el caso de existir alguno (una lámpara por ejemplo), este mismo elemento de referencia constituye un obstáculo en la recopilación de información de la superficie oculta tras él.

# 6. Conclusiones

Como se ha podido comprobar, se ha conseguido una obtención parcial de los objetivos propuestos en un principio.

El proceso global de obtención de resultados ha demostrado ser el apropiado para la mayoría de los casos, si bien ha presentado ciertas dificultades para ciertos elementos, en concreto aplicado a los entornos más extensos.

Se ha dado prioridad a la forma de capturar los datos, intentando obtener unas Point Clouds iniciales que fuesen lo suficientemente representativas como para asegurar un correcto funcionamiento del programa de procesado.

Considerando las fuentes de error y los resultados, quizá hubiese sido más conveniente poner más énfasis en este proceso de tratamiento, de manera que fuese posible extraer las superficies regulares sin necesitar que fuese un requerimiento indispensable que el estado de la Point Cloud no presentase más que pequeños errores a la hora de aplicarle el algoritmo. Este aspecto se ve corroborado especialmente por los datos post-procesamiento obtenidos del Dormitorio 4 y del Hall, que a su vez generan una vía de trabajo complementaria a este proyecto y más centrada en la optimización de los contornos finales obtenidos.

Sin duda, la variación de parámetros del software ha conseguido mejorar la precisión de las nubes de puntos y adecuar el proceso al hardware utilizado, pero no ha sido suficiente para asegurar la exactitud de los modelos finales asociados a cada habitación.

El proceso SLAM, si bien ha demostrado su utilidad y capacidad para generar mapas tridimensionales y estimaciones de trayectorias adecuadas, involucra una componente de aleatoriedad que en el caso de estancias más grandes, como ha sido el caso, la posibilidad de aparición de algún error aumente, influyendo en los datos de entrada de la parte de tratamiento.

Por otro lado, se ha demostrado la utilidad de una matriz topogeométrica a la hora de proporcionar al robot una información que le permita generar una trayectoria eficiente entre 2 secciones cualesquiera del mapa global.

# 7. Costes del proyecto

Tabla 8 - Tabla de costes

| **Concepto** | **Importe** |
|---|---|
| Sensor ASUS Xtion  Pro Live | 139 € |
| Horas trabajadas*: | 3750 € |
| SO Ubuntu 12.04 Precise | - € |
| Software RGBD-SLAM | - € |
| PCL | - € |
| **TOTAL** | **3889 €** |

:

**Fórmula aplicada: Coste=Nº horas trabajadas * Tarifa horaria estimada + Nº horas trabajadas con el tutor * Tarifa horaria estimada =300* 10 €/h + 15 * 50 € /h*

*Horas trabajadas : 300 horas*

*Horas con el tutor : 15 horas*

# 8. Trabajos futuros y mejoras propuestas

Dejando a un lado la viabilidad de utilizar los datos específicos de este trabajo como punto de partida de futuros proyectos. La idea considerada en los objetivos del mismo sí que constituye un elemento que permite enlazarlo con aplicaciones orientadas a distintos procesos del proyecto.

Para comenzar, hay una gran cantidad de funcionalidades que pueden ser agregadas al software de tratamiento de Point Clouds. La librería PCL proporciona unas herramientas para optimizar los procesos ya incluidos y añadir otros nuevos.

Una opción especialmente útil sería la de definir la linealidad del contorno de la imagen en el proceso de extracción del mismo. Una vez obtenida la Point Cloud del contorno, se podría estimar el conjunto de puntos que representan la distribución real de la recta de la pared y aplicar una traslación a los mismos para que cumplan la solución estimada.

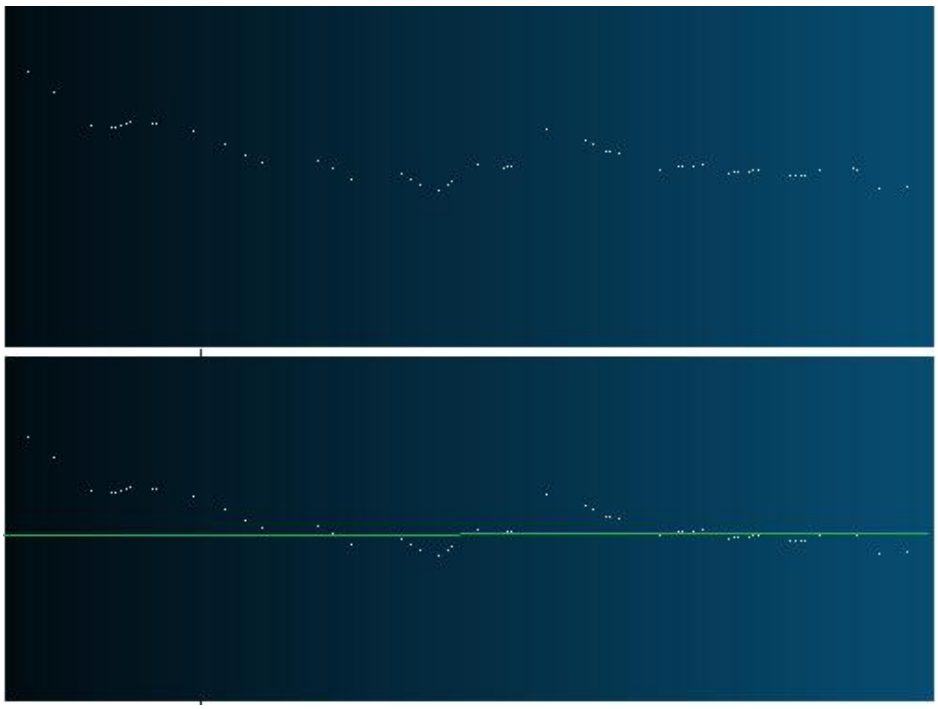


Ilustración 53 – Posible recta solución a la linealidad del contorno

Por otra parte, este proceso no solo sería útil para paliar el posible efecto de errores en la captura, sino también para eliminar de la representación del contorno elementos como estanterías, cuadros, etc. que si bien son deseados para la representación del

modelo 3D local, ya se ha establecido que no tienen ninguna utilidad de cara al mapa topogeométrico.

En otro rango de aplicaciones, hay que considerar el empleo real de un mapa de estas características en un sistema SLAM integrado en un robot.

Hay muchísimas posibilidades en cuanto al uso que un robot podría hacer si dispusiese del mapa completo (más o menos optimizado) en tres dimensiones de la estancia en la que se encuentra. La combinación de poseer la geometría de los objetos de una habitación y un algoritmo de cálculo de los puntos de agarre de los mismos permitiría seleccionar al robot mejor equipado para una tarea específica previamente.

Completando la introducción de la solución del problema SLAM mediante el uso de un mapa global capturado en un solo recorrido del robot, se podría realizar un análisis de la factibilidad de realizar un tratamiento a estos mapas globales para segmentarlos en diferentes estancias. Se trataría de llegar al mismo resultado que este proyecto pero partiendo de un único mapa global, y aplicando ciertos criterios como:

- Identificación de puerta (discontinuidades en el contorno
- Identificación de espacios cerrados (superficies cerradas en un %)

con el fin de obtener un mapa topogeométrico como el buscado en este trabajo.

Por último, y teniendo en cuenta los problemas asociados a zonas “vacías” en el mapa topogeométrico, ya sea por un dimensionamiento erróneo de alguno de los contorno o por la ausencia de información en ciertas regiones en otros, otra propuesta de tratamiento posterior a la generación del mapa sería la de rellenar estas zonas oscuras.

En el caso del Hall en este proyecto, el algoritmo debería identificar las rectas del contorno de los Dormitorios 1 y 2 para poder cerrar el grafo en la región situada a la derecha.

# 9. Bibliografía

[1] Wolfram Burgard, Cyrill Stachniss,Kai Arras, Maren Bennewitz - "SLAM: Simultaneous Localization and Mapping"

[2] Hugh Durrant-Whyte, Fellow, IEEE, and Tim Bailey - "Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms"

[3] Simultaneous Localizationand Mapping (SLAM):Part II BY TIM BAILEY AND HUGH DURRANT-WHYT

[4] Apuntes de Introducción al SLAM. Disponibles en: http://www.comp.dit.ie/jkelleher/rtf/classmaterial/week8/MappingAndSLAM.pdf

[5] Giorgio Grisetti, Andreas Nuechter, Alexander Kleiner – "Tutorial: SLAM to the Rescue. A hands-on tutorial on using state-of-the-art SLAM algorithms in harsh environments", 2011.

[6] Nuchter, A. ; Fraunhofer Inst. for Autonomous Intelligent Syst., Sankt Augustin, Germany ; Surmann, H. ; Lingemann, K. ; Hertzberg, J. – "6D SLAM with an application in autonomous mine mapping"

[7] Luc Jaulin1, Frédéric Dabe2, Alain Bertholom2, and Michel Legris - "A set approach to the simultaneous localization and map building; application to underwater robots"

[8] Margarita Chli - Autonomous Systems Lab, ETH Zurich "Visual SLAM-for small Unmanned Aerial Vehicles"

[9] Frank Steinbrücker/ Jürgen Sturm /Daniel Cremers – "Real-Time Visual Odometry from Dense RGB-D Images"

[10] Thomas Whelan, Hordur Johannsson, Michael Kaess, John J. Leonard and John McDonald – "Robust Real-Time Visual Odometry for Dense RGB-D Mapping"

[11] http://www.ubuntu.com/desktop

[12] http://www.geany.org/Main/HomePage

[13] http://www.cmake.org/

[14] http://www.asus.com/Multimedia/Xtion_PRO_LIVE/specifications

[15] Krystof Litomisky – "Consumer RGB-D Cameras and their Applications" . Universidad of California, Riverside. Spring 2012.

[16] Felix Endres, Jürgen Hess, Nikolas Engelhard, Jürgen Sturm, Daniel Cremers, Wolfram Burgard – "An Evaluation of the RGB-D SLAM System"

[17] Dr. Francis Colas. Presentación "Iterative Closest Point Algorithm". Disponible en: http://www.asl.ethz.ch/education/master/info-process-rob/ICP.pdf

[18] Michael Wild – "Recent development of the Iterative Close Point (ICP) Algorithm - An Overwiew of the Years 2002 to 2007". Supervised by: Francois Pomerleau and Francis Colas. Autumn Term 2010.

[19] Robert B. Fisher –" The RANSAC (Random Sample Consensus) Algorithm"

[20] Konstatinos G.Derpanis – "Overview of the RANSAC Algorithm". May 2010

[21] http Armin Hornung – Presentación "3D Mapping with OctoMap". Joint work with K.M. Wurm, M. Bennewitz, C. Stachniss, W. Burgard. Disponible en: http://www2.informatik.uni-freiburg.de/~hornunga/pub/hornung13roscon.pdf

[22] A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard: "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees" in Autonomous Robots Vol 34, 2013

[23] M.Y.I. Idris, H. Arof, E.M. Tamil, N.M. Noor and Z. Razak, 2009. Review of Feature Detection Techniques for Simultaneous Localization and Mapping and System on Chip Approach. Information Technology Journal, 8: 250-262.

[24] Formato pcd - http://pointclouds.org/documentation/tutorials/pcd_file_format.php

[25] Página web donde se encuentran varias de las soluciones SLAM más importantes en la actualidad, entre ellas el software RGBDSLAM - www.openslam.org

# 10. ANEXO I – Manual de Usuario para la instalación de RGBDSLAM

**Instalación de ROS-fuerte (Ubunto 12.04 Precise) y creación del espacio de trabajo**

sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu precise main" > /etc/apt/sources.list.d/ros-latest.list'

wget http://packages.ros.org/ros.key -O - | sudo apt-key add -
sudo apt-get update
sudo apt-get install ros-fuerte-desktop-full
sudo apt-get install python-rosinstall python-rosdep

rosws init ~/fuerte_workspace /opt/ros/fuerte

//Define tu ROS_PACKAGE_PATH con

export ROS_PACKAGE_PATH=~/fuerte_workspace:$ROS_PACKAGE_PATH

//Direcciona los nuevos “packages” al espacio de trabajo recién creado

gedit ~/.bashrc

//Añade la siguiente línea al final del archivo y GUARDA
source ~/fuerte_workspace/setup.bash

**Instalación de rosdep**

sudo apt-get install python-rosdep
sudo rosdep init
rosdep update

**Instalación de RGBDSLAM**

cd fuerte_workspace
svn co http://alufr-ros-pkg.googlecode.com/svn/trunk/rgbdslam_freiburg
rosws set rgbdslam_freiburg
rosdep update
rosdep install rgbdslam_freiburg
rosmake rgbdslam_freiburg

**Complementos**

sudo apt-get install libglew1.5-dev libdevil-dev libsuitesparse-dev
sudo apt-get install ros-fuerte-openni-launch